using System;

using System.Collections.Generic;

namespace FurnitureStore

{

public interface IAssemble

{

void Assemble();

void ShowInfo();

}

public enum MaterialType

{

Wood,

Metal,

Leather,

Fabric

}

public struct Dimensions

{

public double Width;

public double Height;

public Dimensions(double width, double height)

{

Width = width;

Height = height;

}

public override string ToString()

{

return $"{Width}x{Height} см";

}

}

public abstract class Product

{

public string Name;

public decimal Price;

public double Weight;

public Product(string name, decimal price, double weight)

{

Name = name;

Price = price;

Weight = weight;

}

public abstract void DisplayInfo();

public abstract void ShowInfo();

public override string ToString()

{

return $"{Name}, Цена: {Price} руб., Вес: {Weight} кг";

}

}

public class Sofa : Furniture

{

public int SeatCount;

public Sofa(decimal price, double weight, MaterialType material, string color, int seatCount, Dimensions size)

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

{

SeatCount = seatCount;

}

public override void DisplayInfo()

{

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

}

}

public class Bed : Furniture

{

public string BedSize;

public Bed(decimal price, double weight, MaterialType material, string color, string bedSize, Dimensions size)

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

{

BedSize = bedSize;

}

public override void DisplayInfo()

{

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

}

}

public class Hanger : Product, IAssemble

{

public string Type;

public Hanger(decimal price, double weight, string type)

: base("Вешалка", price, weight)

{

Type = type;

}

public override void DisplayInfo()

{

Console.WriteLine($"{Name}: {Type}, Цена: {Price} руб., Вес: {Weight} кг");

}

public void Assemble()

{

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

}

public override void ShowInfo()

{

Console.WriteLine($"Общая информация - Вешалка: Тип: {Type}, Цена: {Price} руб., Вес: {Weight} кг");

}

}

public class Warehouse<T> where T : Product //Обобщённые классы позволяют создавать код, работающий с разными типами данных без дублирования кода.

{

// Поле для хранения списка продуктов

private List<T> products = new List<T>();//Это поле представляет собой список объектов типа T, которые хранятся на складе. Список — это динамическая коллекция, которая позволяет добавлять, удалять и хранить элементы.

// Свойство для получения и установки списка продуктов

public List<T> Products //Getter (get) — возвращает текущий список продуктов.

//Setter(set) — позволяет установить новый список продуктов, если это нужно.

{

get { return products; }

set { products = value; }

}

// Метод для добавления продукта в список

public void AddProduct(T product)

{

products.Add(product);

}

// Метод для удаления продукта из списка

public void RemoveProduct(T product)

{

products.Remove(product);

}

// Метод для вывода всех продуктов на консоль

public void DisplayProducts()

{

if (products.Count == 0)

{

Console.WriteLine("Список продуктов пуст.");

return;

}

foreach (var product in products)

{

product.ShowInfo(); // Вывод информации о каждом продукте

}

}

public IEnumerable<T> FindByManufacturer(string manufacturer)//В результате метод будет возвращать коллекцию продуктов, которые соответствуют условиям поиска.IEnumerable — это интерфейс, который позволяет перебор коллекций

{

// Создаём временную коллекцию для хранения отфильтрованных продуктов

List<T> foundProducts = new List<T>();

// Проходим по каждому продукту в списке

foreach (T product in products)

{

// Проверяем, содержит ли имя продукта строку manufacturer

if (product.Name.Contains(manufacturer))

{

// Если содержит, добавляем продукт в временную коллекцию

foundProducts.Add(product);

}

}

// Возвращаем отфильтрованную коллекцию

return foundProducts;

}

public decimal GetTotalPriceByType(Type type)

{

decimal totalPrice = 0; // Переменная для хранения общей суммы

// Проходим по каждому продукту в списке

foreach (T product in products)

{

// Если тип продукта совпадает с переданным типом

if (product.GetType() == type)

{

// Добавляем цену продукта к общей сумме

totalPrice += product.Price;

}

}

// Возвращаем общую сумму

return totalPrice;

}

public List<T> GetProducts()

{

return products; // Просто возвращаем список продуктов

}

}

public class PriceComparer : IComparer<Product> //IComparer<T> — это интерфейс в .NET, который используется для создания пользовательских алгоритмов сравнения объектов. Он имеет один метод Compare(T x, T y), который сравнивает два объекта типа T и возвращает:

{

public int Compare(Product x, Product y)

{

if (x == null || y == null) return 0;

return x.Price.CompareTo(y.Price);//Здесь вызывается метод CompareTo для свойства Price объекта x, который сравнивается с ценой объекта y:

}

}

public class WeightComparer : IComparer<Product>

{

public int Compare(Product x, Product y)

{

if (x == null || y == null) return 0;

return x.Weight.CompareTo(y.Weight);

}

}

public class WarehouseController

{

private Warehouse<Product> warehouse;// warehouse — это объект класса Warehouse, который работает с продуктами типа Product.

public WarehouseController(Warehouse<Product> warehouse) //

{

this.warehouse = warehouse;//Сохрани переданный мне склад (параметр warehouse) в поле warehouse, которое принадлежит этому контроллеру (this.warehouse)."

}

public void AddProduct(Product product)

{

warehouse.AddProduct(product);

}

public void RemoveProduct(Product product)

{

warehouse.RemoveProduct(product);

}

public void ShowAllProducts()

{

foreach (var product in warehouse.GetProducts())

{

product.ShowInfo();

}

}

public void ShowProductsByManufacturer(string manufacturer)

{

var products = warehouse.FindByManufacturer(manufacturer);

foreach (var product in products)

{

product.ShowInfo();

}

}

public void SortProductsByPrice()

{

var sortedProducts = warehouse.GetProducts();//вызываем метод GetProducts у объекта warehouse, который возвращает список всех продуктов на складе.

sortedProducts.Sort(new PriceComparer());//Здесь мы создаём новый объект класса PriceComparer. Этот класс реализует интерфейс IComparer<Product>, который требует определения метода Compare. Метод Compare определяет, как сравнивать два объекта типа Product на основе их цен.

foreach (var product in sortedProducts)

{

product.ShowInfo();

}

}

public void SortProductsByWeight()

{

var sortedProducts = warehouse.GetProducts();

sortedProducts.Sort(new WeightComparer());

foreach (var product in sortedProducts)

{

product.ShowInfo();

}

}

public void ShowTotalPriceByType(Type type)

{

decimal totalPrice = warehouse.GetTotalPriceByType(type);

Console.WriteLine($"Общая стоимость товаров типа {type.Name}: {totalPrice} руб.");

}

}

class Program

{

static void Main()

{

Dimensions sofaDimensions = new Dimensions(200, 90);

Dimensions sofatwoDimensions = new Dimensions(300, 100);

Dimensions bedDimensions = new Dimensions(210, 110);

Dimensions bedDimensionstwo = new Dimensions(230, 140);

var sofa = new Sofa(499, 90, MaterialType.Leather, "Чёрный", 3, sofaDimensions);

var sofatwo = new Sofa(599, 80, MaterialType.Leather, "Белый", 3, sofatwoDimensions);

var bed = new Bed(799, 60, MaterialType.Wood, "Белый", "King-size", bedDimensions);

var bedtwo = new Bed(999, 50, MaterialType.Wood, "Чёрный", "King-size", bedDimensionstwo);

var hanger = new Hanger(15, 1, "Настенная");

Warehouse<Product> warehouse = new Warehouse<Product>();

WarehouseController controller = new WarehouseController(warehouse);

controller.AddProduct(sofa);

controller.AddProduct(sofatwo);

controller.AddProduct(bed);

controller.AddProduct(bedtwo);

controller.AddProduct(hanger);

Console.WriteLine("Все товары на складе:");

controller.ShowAllProducts();

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

controller.ShowTotalPriceByType(typeof(Sofa));

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

Console.WriteLine("\nТовары от производителя 'Кровать':");

controller.ShowProductsByManufacturer("Кровать");

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

Console.WriteLine("\nТовары после сортировки по цене:");

controller.SortProductsByPrice();

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

Console.WriteLine("\nТовары после сортировки по весу:");

controller.SortProductsByWeight();

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

controller.RemoveProduct(sofa);

controller.RemoveProduct(sofatwo);

Console.WriteLine("Диваны удалены со склада, список после удаления:");

controller.ShowAllProducts();

}

}

}