// Products.cs

using System;

namespace Task1

{

public class Product

{

private string name = "Unnamed";

private double cost = 0.0;

private double weight = 0.0;

public string Name

{

get => name;

set

{

if (value == null)

{

throw new ArgumentNullException(nameof(value), $"{nameof(Name)} cannot be set to null.");

}

if (string.Compare(value, "") == 0)

{

throw new ArgumentException($"{nameof(value)} is empty string.", $"{nameof(Name)} cannot be set to empty string.");

}

name = value;

}

}

public double Cost

{

get => cost;

set

{

if (value <= 0.0)

{

throw new ArgumentException($"{nameof(value)} is lower than 0.0", $"{nameof(Cost)} cannot be set to negative number.");

}

cost = value;

}

}

public double Weight

{

get => weight;

set

{

if (value < 0.0)

{

throw new ArgumentException($"{nameof(value)} is lower than 0.0", $"{nameof(Weight)} cannot be set to negative number.");

}

weight = value;

}

}

public Product(string name, double cost, double weight)

{

this.Name = name;

this.Cost = cost;

this.Weight = weight;

}

public Product(Product product) : this(product.Name, product.Cost, product.Weight) { }

public override bool Equals(object obj)

{

if (obj.GetType() != this.GetType())

{

return false;

}

var other = obj as Product;

return String.Compare(this.Name, other.name) == 0 &&

this.Cost == other.Cost &&

this.Weight == other.Weight;

}

public override string ToString()

{

return String.Concat($"Name: {this.Name}\n", $"Cost: ${this.Cost}\n", $"Weight: {this.Weight} kg");

}

public override int GetHashCode()

{

return HashCode.Combine(this.Name, this.Cost, this.Weight);

}

}

}

// Buy.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task1

{

public class Buy

{

private List<Product> products;

public List<Product> Products

{

get => products;

}

public int Count

{

get => Products.Count;

}

public double OverallCost

{

get => Products.Sum(product => product.Cost);

}

public double OverallWeight

{

get => Products.Sum(product => product.Weight);

}

public Buy(params Product[] products)

{

this.products = new List<Product>();

for (int i = 0; i < products.Length; ++i)

{

if (products[i] == null)

{

throw new ArgumentNullException($"products[{i}] is null.");

}

Products.Add(new Product(products[i]));

}

}

public override string ToString()

{

StringBuilder sb = new();

sb.Append($"Count: {Count}\n");

sb.Append(String.Join("\n", Products.Select((product, index) => $"Product #{index + 1}:\n{product.ToString()}")));

sb.Append($"\nOverall weight: {OverallWeight}\n");

sb.Append($"Overall cost: {OverallCost}\n");

return sb.ToString();

}

public override bool Equals(object obj)

{

if (obj.GetType() != this.GetType())

{

return false;

}

var other = obj as Buy;

return this.Count == other.Count &&

this.OverallWeight == other.OverallWeight &&

this.OverallCost == other.OverallCost &&

this.Products.Equals(other.Products);

}

public override int GetHashCode()

{

return HashCode.Combine(this.Count, this.OverallCost, this.OverallWeight, this.Products);

}

}

}

// Check.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task1

{

public class Check

{

private Check() { }

public static string PrintProduct(Product product)

{

if (product == null)

{

throw new ArgumentNullException(nameof(product));

}

return product.ToString();

}

public static string PrintBuy(Buy buy)

{

if (buy == null)

{

throw new ArgumentNullException(nameof(buy));

}

return buy.ToString();

}

}

}

// Program.cs

using System;

namespace Task1

{

class Program

{

static void Main(string[] args)

{

DemonstrateMethods();

}

static void DemonstrateMethods()

{

Product product = new("NVidia GeForce RTX 3090", 799, 0.8);

Buy buy = new(product, new Product("Intel Core i9-11900K", 500, 0.5));

Console.WriteLine(Check.PrintBuy(buy));

Console.WriteLine(product.ToString());

}

}

}