using System;

using System.Collections.Generic;

namespace Mediator

{

enum Products

{

Chocolate,

Toy,

Computer

}

class ProductsInfo

{

public Products ProductType { get; set; }

public int ProductNumber { get; set; }

public double ProductPrice { get; set; }

public ProductsInfo() { }

public ProductsInfo(Products product, int number, double price)

{

ProductType = product;

ProductNumber = number;

ProductPrice = price;

}

}

abstract class AllYouNeed

{

protected abstract Products \_productType { get; set; }

protected abstract string \_name { get; set; }

protected abstract int \_number { get; set; }

protected abstract double \_price { get; set; }

public abstract void BuyProduct();

public abstract void SellProduct(Products product, int number, double price);

public abstract void AddRegister();

public abstract void RemoveRegister();

}

class Factory : AllYouNeed

{

protected override Products \_productType { get; set; }

protected override int \_number { get; set; }

protected override double \_price { get; set; }

protected override string \_name { get; set; }

public Factory(string factoryName, Products productType, int number = 0, double price = 0)

{

\_name = factoryName;

\_productType = productType;

\_number = number;

\_price = price;

}

public override void BuyProduct() => Console.WriteLine("this method does not work here. Please call the Mediator.");

public override void SellProduct(Products product, int number, double price)

{

Program.Mediator.SellProduct(\_name, \_productType, number, price);

\_number -= number;

}

public override void AddRegister() => Program.Mediator.AddProducter(\_name, \_productType);

public override void RemoveRegister() => Program.Mediator.RemoveProducter(\_name, \_productType);

}

class Store : AllYouNeed

{

protected override Products \_productType { get; set; }

protected override int \_number { get; set; }

protected override double \_price { get; set; }

protected override string \_name { get; set; }

public Store(string name, Products productType, int number = 0, double price = 0)

{

\_name = name;

\_productType = productType;

\_number = number;

\_price = price;

}

public override void AddRegister() => Program.Mediator.AddStoreer(\_name, \_productType);

public override void BuyProduct()

{

var isTrue = Program.Mediator.ProductSalesShow(\_productType);

if (isTrue)

{

Console.Write("Select product: ");

int select = int.Parse(Console.ReadLine());

var infoProduct = Program.Mediator.BuyProducts(select, \_name, \_productType);

this.\_number += infoProduct.ProductNumber;

this.\_price += infoProduct.ProductPrice;

}

}

public override void RemoveRegister() => Program.Mediator.RemoveStoreer(\_name, \_productType);

public override void SellProduct(Products product = Products.Toy, int number = 0, double price = 0) => Console.WriteLine("this method does not work here. Please call the Mediator.");

}

class Mediator

{

private Dictionary<Products, string> \_customersFactory;

private Dictionary<Products, string> \_customersStore;

private List<ProductsInfo> \_productsInfos;

public Mediator()

{

\_customersFactory = new Dictionary<Products, string>();

\_customersStore = new Dictionary<Products, string>();

\_productsInfos = new List<ProductsInfo>();

}

public ProductsInfo BuyProducts(int index, string storeName, Products productName)

{

Products? product = null;

int counter = 0;

foreach (var store in \_customersStore)

{

if (storeName == store.Value && productName == store.Key)

{

product = store.Key;

break;

}

}

if (product != null)

{

index--;

if (index < \_productsInfos.Count)

{

var temporaryRetainer = \_productsInfos[index];

\_productsInfos.RemoveAt(index);

Console.WriteLine($"{temporaryRetainer.ProductType}, {temporaryRetainer.ProductNumber}, {temporaryRetainer.ProductPrice}. Product was sold.");

return temporaryRetainer;

}

else

{

Console.WriteLine($"Product don't was sold.");

return new ProductsInfo();

}

}

else

{

Console.WriteLine("You aren't registered.");

return null;

}

}

public void SellProduct(string Name, Products productName, int number, double price)

{

Products? productt = null;

int counter = 0;

foreach (var customer in \_customersFactory)

{

if (Name == customer.Value && productName == customer.Key)

{

productt = customer.Key;

break;

}

}

if (productt != null)

{

\_productsInfos.Add(new ProductsInfo(productName, number, price));

Console.WriteLine("The product went on sale.");

}

else

Console.WriteLine("Don't the product went on sale. You did not register us.");

}

public bool ProductSalesShow(Products productType)

{

if (\_productsInfos.Count == 0)

{

Console.WriteLine("Products that match your product type are not sold.");

return false;

}

else

{

Console.WriteLine("\tID\t|\tProduct Type\t|\tNumber\t|\tPrice\t|");

for (int i = 0; i < \_productsInfos.Count; i++)

{

if (\_productsInfos[i].ProductType == productType)

{

if (productType == Products.Toy)

Console.WriteLine($"\t{i + 1}\t|\t{\_productsInfos[i].ProductType}\t\t|\t{\_productsInfos[i].ProductNumber}\t|\t{\_productsInfos[i].ProductPrice}\t|");

else

Console.WriteLine($"\t{i + 1}\t|\t{\_productsInfos[i].ProductType}\t|\t{\_productsInfos[i].ProductNumber}\t|\t{\_productsInfos[i].ProductPrice}\t|");

}

}

Console.WriteLine("=================================================================================");

return true;

}

}

public void AddProducter(string factoryName, Products productName)

{

if (\_customersFactory.Count == 0)

{

\_customersFactory.Add(productName, factoryName);

Console.WriteLine("You have successfully registered.");

}

else

{

bool isCorrect = false;

foreach (var customer in \_customersFactory)

{

if (factoryName == customer.Value && productName == customer.Key)

{

isCorrect = false;

break;

}

else

isCorrect = true;

}

if (isCorrect)

{

\_customersFactory.Add(productName, factoryName);

Console.WriteLine("You have successfully registered.");

}

else

Console.WriteLine("There is already a company on this island.");

}

}

public void RemoveProducter(string factoryName, Products productName)

{

if (\_customersFactory.Count == 0)

Console.WriteLine("You aren't registered.");

else

{

Products? product = null;

int counter = 0;

foreach (var customer in \_customersFactory)

{

if (factoryName == customer.Value && productName == customer.Key)

{

product = customer.Key;

break;

}

}

if (product != null)

{

\_customersFactory.Remove((Products)product);

Console.WriteLine("We wish you good luck.");

}

else

Console.WriteLine("You aren't registered.");

}

}

public void AddStoreer(string storeName, Products productName)

{

if (\_customersStore.Count == 0)

{

\_customersStore.Add(productName, storeName);

Console.WriteLine("You have successfully registered.");

}

else

{

bool isCorrect = false;

foreach (var store in \_customersStore)

{

if (storeName == store.Value && productName == store.Key)

{

isCorrect = false;

break;

}

else

isCorrect = true;

}

if (isCorrect)

{

\_customersStore.Add(productName, storeName);

Console.WriteLine("You have successfully registered.");

}

else

Console.WriteLine("There is already a company on this island.");

}

}

public void RemoveStoreer(string storeName, Products productName)

{

if (\_customersStore.Count == 0)

Console.WriteLine("You aren't registered.");

else

{

Products? product = null;

int counter = 0;

foreach (var store in \_customersStore)

{

if (storeName == store.Value && productName == store.Key)

{

product = store.Key;

break;

}

}

if (product != null)

{

\_customersStore.Remove((Products)product);

Console.WriteLine("We wish you good luck.");

}

else

Console.WriteLine("You aren't registered.");

}

}

}

internal class Program

{

public static Mediator Mediator = new Mediator();

static void Main(string[] args)

{

// no registered.

Factory factory1 = new Factory("ChocolateFactory", Products.Chocolate);

Factory factory2 = new Factory("ToyFactory", Products.Toy);

Factory factory3 = new Factory("ComputerFactory", Products.Computer);

Store store1 = new Store("ComputerStore", Products.Computer);

Store store2 = new Store("ChocolateStore", Products.Chocolate);

Store store3 = new Store("ToyStore", Products.Toy);

factory1.SellProduct(Products.Chocolate, 1, 1.5);

factory2.SellProduct(Products.Chocolate, 2, 2.5);

factory3.SellProduct(Products.Chocolate, 3, 3.5);

Console.WriteLine();

Console.WriteLine();

store1.BuyProduct();

store2.BuyProduct();

store3.BuyProduct();

Console.WriteLine();

Mediator.ProductSalesShow(Products.Chocolate);

Console.WriteLine();

Console.WriteLine();

Console.WriteLine();

// --------------------------------------------------------------------------------

// yes registered and add info and info show.

factory1.AddRegister();

factory2.AddRegister();

factory3.AddRegister();

Console.WriteLine();

Console.WriteLine();

store1.AddRegister();

store2.AddRegister();

store3.AddRegister();

factory1.SellProduct(Products.Chocolate, 1, 1.5);

factory2.SellProduct(Products.Chocolate, 2, 2.5);

factory3.SellProduct(Products.Chocolate, 3, 3.5);

Console.WriteLine();

Console.WriteLine();

Mediator.ProductSalesShow(Products.Chocolate);

Console.WriteLine();

Console.WriteLine();

Mediator.ProductSalesShow(Products.Computer);

Console.WriteLine();

Console.WriteLine();

Mediator.ProductSalesShow(Products.Toy);

// --------------------------------------------------------------------------------

// not working methods.

factory1.BuyProduct();

factory2.BuyProduct();

factory3.BuyProduct();

Console.WriteLine();

Console.WriteLine();

store1.SellProduct();

store2.SellProduct();

store3.SellProduct();

}

}

}

//Mediator, obyektlər arasındakı xaotik asılılıqları azaltmağa imkan verən davranış tərzidir. Nümunə, obyektlər arasında birbaşa əlaqəni məhdudlaşdırır və onları yalnız vasitəçi obyekt vasitəsilə əməkdaşlıq etməyə məcbur edir.