МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа №2

Магазин

Выполнил студент группы М3212:

Муртазин Рифат

САНКТ-ПЕТЕРБУРГ

2020

Задание

Есть **Товары**, которые продаются в **Магазинах**.

У **магазинов** есть код (уникальный), название (не обязательно уникальное) и адрес.

У **товаров** есть код (уникальный), название (не обязательно уникальное).

В каждом магазине установлена своя цена на товар и есть в наличии некоторое количество единиц товара (какого-то товара может и не быть вовсе).

Написать методы для следующих операций:

1. Создать магазин;
2. Создать товар;
3. Завезти партию товаров в магазин (набор товар-количество с возможностью установить/ изменить цену);
4. Найти магазин, в котором определенный товар самый дешевый;
5. Понять, какие товары можно купить в магазине на некоторую сумму (например, на 100 рублей можно купить три кг огурцов или две шоколадки);
6. Купить партию товаров в магазине (параметры - сколько каких товаров купить, метод возвращает общую стоимость покупки либо её невозможность, если товара не хватает);
7. Найти, в каком магазине партия товаров (набор товар-количество) имеет наименьшую сумму (в целом). Например, «в каком магазине дешевле всего купить 10 гвоздей и 20 шурупов». Наличие товара в магазинах учитывается!

Для демонстрации необходимо создать минимум 3 различиных магазина, 10 типов товаров и наполнить ими магазины.

Ход рассуждений

Магазин состоит из 5 классов: **Product, ProductInStore, ShopManager, Store и Store_Exception.**

Класс **Product** имеет 2 свойства: productID и productName. Класс **ProductInStore** описывает продукт, который находится в магазине и имеет цену и количество, класс наследуется от класса Product. Класс **Store** описывает магазин и он имеет 4 свойства: Id, имя, адресс и словарь всех продуктов в нём. Также имеет методы, при помощи которых можно взаимодействовать с магазином. Можно добавить продукты в магазин при помощи add_product, поменять ценник и количество товара в магазине методами change_price и change_quantity, проверить существование товара в магазине методом check_product_in_list, получить список продуктов, которые можно купить на определённую сумму методом available_product и получить общую стоимость покупки при помощи метода get_sell_amount_cost. Класс **ShopManager** выполняет функцию взаимодействия со всеми магазинами. Он хранит их в словаре StoresList. Также этот класс хранит в словаре AllProductsList все существующие продукты, которые были созданы, но не завезены в конкретный магазин. В данном классе можно создать магазин при помощи метода make_product, можно сделать проверки на существование магазина или продукта методами try_store_list_ID и try_product_list_ID. Также, если хочется найти в каком магазине выгоднее купить, какой-то товар или товары, можно вызвать метод cheap_product_store или best_store, которые подскажут, в каком магазине лучше купить, например бутылку воды.

В **Program.cs** выполняются все проверки. Вначале создаю экземпляр класса ShopManager, затем создаю 3 магазина, в которые добавляю по 11 товаров, но перед добавлением создаю сами товары и только после этого завожу в сами магазины. После меняю цену, количество для проверки работы функционала. Далее вывожу для каждого магазина список продуктов с количеством, ценой, названием и ID. Потом проверяю для каждого магазина, что я могу купить на 1000 у.е. и вывожу результаты. Затем наполняю тележку продуктами, у каждого магазина она своя. И, как аргумент использую, вызывая метод, который считает общую стоимость товаров, которые лежат в магазине.

При возникновении исключений предусмотрён класс Store_Exception, который просто наследуется от Exception и передаёт ему свои сообщения. Также в программе стоит секундомер, который высчитывет время выполнения блока try. Вставлен для отслеживания времени выполнения программы.

Листинг

Файл Product.cs

```
using System;
namespace Code_of_lab_2
{
   public class Product
   {
      public string productID   { protected set; get; }
      public string productName { protected set; get; }

      public Product(string productName)
      {
         this.productID = Guid.NewGuid().ToString();
         this.productName = productName;
      }

   }
}
```

Файл Store.cs

```
using System;
using System.Collections.Generic;

namespace Code_of_lab_2
{
   public class Store
   {
      public string storeID      { private set; get; }
      public string storeName    { private set; get; }
      public string storeAddress { private set; get; }
      public Dictionary<string, ProductInStore> AllProductInStore { protected set; get; } = new Dictionary<string, ProductInStore>();

   public Store(string storeName, string storeAddress)
      {
         this.storeID = Guid.NewGuid().ToString();
         this.storeName = storeName;
         this.storeAddress = storeAddress;
      }
      public Store()
      {
      }

      public void add_product(Product currentProdInList, int quantity, double price)
      {
         ProductInStore product = new ProductInStore(currentProdInList, quantity, price);
         AllProductInStore.Add(product.productID, product);
      }

      public void change_price(string ID, double newPrice)
      {
         if (newPrice >= 0)
         {
            AllProductInStore[ID].price = newPrice;
         }
         else
         {
            throw new Store_Exception("Price doesn't be less 0");
         }

      }
      public void change_quantity(string ID, int newQuantity)
      {
         if (newQuantity >= 0)
         {
            AllProductInStore[ID].quantity = newQuantity;
         }
         else
         {
            throw new Store_Exception("Quantity doesn't be less 0");
```

```
    }

}

public bool check_product_in_list(string ID)
{
    return AllProductInStore.ContainsKey(ID);
}

public List<KeyValuePair<ProductInStore, int>> available_product(double money)
{
    List<KeyValuePair<ProductInStore, int>> AvailableProduct = new List<KeyValuePair<ProductInStore, int>>();
    int quantity = 0;
    foreach (KeyValuePair<string, ProductInStore> keyValue in AllProductInStore)
    {
        if (keyValue.Value.price < money && money / keyValue.Value.price < keyValue.Value.quantity)
        {
            quantity = (int)(money / keyValue.Value.price);
            AvailableProduct.Add(new KeyValuePair<ProductInStore, int>(keyValue.Value, quantity));
        }
        else if (money / keyValue.Value.price > keyValue.Value.quantity)
        {
            quantity = keyValue.Value.quantity;
            AvailableProduct.Add(new KeyValuePair<ProductInStore, int>(keyValue.Value, quantity));
        }
        else
        {
            AvailableProduct.Add(new KeyValuePair<ProductInStore, int>(keyValue.Value, 0));
        }
    }

    return AvailableProduct;
}

public double get_sell_amount_cost(Dictionary<string, int> Cart)
{
    double amount_cost = 0;
    foreach (KeyValuePair<string, int> keyValue in Cart)
    {
        if (keyValue.Value <= AllProductInStore[keyValue.Key].quantity)
        {
            amount_cost += keyValue.Value * AllProductInStore[keyValue.Key].price;
        }
        else
        {
            throw new Store_Exception("Out of stock!");
        }
    }

    return amount_cost;
}
```

```
    }
}
```

Файл Store_Exception.cs

```
using System;
namespace Code_of_lab_2
{
    public class Store_Exception : Exception
    {
        public Store_Exception(string message)
            : base(message)
        { }

    }
}
```

Файл ShopManager.cs

```
using System;
using System.Collections.Generic;

namespace Code_of_lab_2
{
    public class ShopManager
    {
        public Dictionary<string, Store> StoresList { protected set; get; } = new Dictionary<string, Store>();
        public Dictionary<string, Product> AllProductsList { protected set; get; } = new Dictionary<string, Product>();

        public string make_store(string Name, string Address)
        {
            Store store = new Store(Name, Address);
            if (!try_store_list_ID(store.storeID))
            {
                StoresList.Add(store.storeID, store);
            }

            return store.storeID;
        }
        public string make_product(string Name)
        {
            Product product = new Product(Name);
            if (!try_product_list_ID(product.productID))
            {
                AllProductsList.Add(product.productID, product);
            }

            return product.productID;
        }

        public bool try_store_list_ID(string ID)
        {
            return StoresList.ContainsKey(ID);
        }
        public bool try_product_list_ID(string ID)
        {
            return AllProductsList.ContainsKey(ID);
        }

        public Store cheap_product_store(string productID)
        {
            double minPrice = double.MaxValue;
            Store best_store = new Store();
            foreach (KeyValuePair<string, Store> keyValue in StoresList)
            {
                double currentCost = keyValue.Value.AllProductInStore[productID].price;
                if (currentCost < minPrice)
                {
```

```
                minPrice = currentCost;
                best_store = keyValue.Value;
            }
        }

        return best_store;
    }

    public Store best_store(Dictionary<string, int> Cart)
    {
        double minAmountCost = double.MaxValue;
        Store best_store = new Store();
        foreach (KeyValuePair<string, Store> keyValue in StoresList)
        {
            double currentCost = keyValue.Value.get_sell_amount_cost(Cart);
            if (currentCost < minAmountCost)
            {
                minAmountCost = currentCost;
                best_store = keyValue.Value;
            }
        }

        return best_store;
    }

  }
}
```

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace Code_of_lab_2
{
   class Program
   {
      static void Main(string[] args)
      {

         Stopwatch stopWatch = new Stopwatch();
         stopWatch.Start();
         try
         {

            ShopManager Shop_Manager = new ShopManager();

            string Sber_ID = Shop_Manager.make_store("Sber Market", "https://sbermarket.ru");
            string Yandex_ID = Shop_Manager.make_store("Yandex Lavka", "https://lavka.yandex");
            string Utkonos_ID = Shop_Manager.make_store("Utkonos", "https://www.utkonos.ru");

            // test for the number of stores. True if is 3.
            Console.WriteLine(Shop_Manager.StoresList.Count);
            /*
             * test for displaying data about stores
             * TRUE if console are
             * ID ... - Sber Market - https://sbermarket.ru
             * ID ... - Yandex Lavka - https://lavka.yandex
             * ID ... - Utkonos - https://www.utkonos.ru
            */
            foreach (KeyValuePair<string, Store> keyValue in Shop_Manager.StoresList)
            {
               Console.WriteLine(keyValue.Key + " - " + keyValue.Value.storeName + " - " + keyValue.Value.storeAddress);
            }

            string tomat_ID = Shop_Manager.make_product("tomatoes");
            string lemon_ID = Shop_Manager.make_product("lemons");
            string bread_ID = Shop_Manager.make_product("bread"); ;
            string milk_ID = Shop_Manager.make_product("milk");
            string tea_ID = Shop_Manager.make_product("tea");
            string coffee_ID = Shop_Manager.make_product("coffee");
            string water_ID = Shop_Manager.make_product("bottle of water");
            string juice_ID = Shop_Manager.make_product("juice");
            string fish_ID = Shop_Manager.make_product("fish");
            string honey_ID = Shop_Manager.make_product("honey");
            string nuts_ID = Shop_Manager.make_product("nuts");

            // test for the number of AllProductsList. True if is 20.
```

```
Console.WriteLine(Shop_Manager.AllProductsList.Count);

// Test true if answer is Sber Market https://sbermarket.ru
Console.Write("Name of store {0}, ", Shop_Manager.StoresList[Sber_ID].storeName);
Console.WriteLine("address {0}", Shop_Manager.StoresList[Sber_ID].storeAddress);

Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[tomat_ID], 58, 153);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[lemon_ID], 79, 126);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[bread_ID], 159, 57);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[milk_ID], 351, 49);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[tea_ID], 58, 360);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[coffee_ID], 456, 540);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[water_ID], 820, 41);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[juice_ID], 43, 63);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[fish_ID], 17, 478);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[honey_ID], 43, 270);
Shop_Manager.StoresList[Sber_ID].add_product(Shop_Manager.AllProductsList[nuts_ID], 43, 231);

Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[tomat_ID], 142, 179);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[lemon_ID], 64, 135);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[bread_ID], 98, 39);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[milk_ID], 102, 68);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[tea_ID], 20, 238);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[coffee_ID], 48, 1380);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[water_ID], 327, 54);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[juice_ID], 81, 98);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[fish_ID], 32, 340);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[honey_ID], 81, 310);
Shop_Manager.StoresList[Yandex_ID].add_product(Shop_Manager.AllProductsList[nuts_ID], 79, 176);

Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[tomat_ID], 76, 149);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[lemon_ID], 101, 97);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[bread_ID], 231, 62);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[milk_ID], 61, 130);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[tea_ID], 19, 360);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[coffee_ID], 256, 760);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[water_ID], 915, 32);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[juice_ID], 54, 124);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[fish_ID], 9, 610);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[honey_ID], 28, 299);
Shop_Manager.StoresList[Utkonos_ID].add_product(Shop_Manager.AllProductsList[nuts_ID], 24, 315);

Console.WriteLine("Better to buy nuts in " + Shop_Manager.cheap_product_store(nuts_ID).storeName);
Console.WriteLine("Better to tea tea in " + Shop_Manager.cheap_product_store(tea_ID).storeName);

Shop_Manager.StoresList[Yandex_ID].change_price(nuts_ID, 400);
Shop_Manager.StoresList[Yandex_ID].change_price(tea_ID, 539);

Console.WriteLine("Better to buy nuts in " + Shop_Manager.cheap_product_store(nuts_ID).storeName);
Console.WriteLine("Better to tea tea in " + Shop_Manager.cheap_product_store(tea_ID).storeName);

Console.WriteLine("List of products in the Sber Market");
```

```
            foreach (KeyValuePair<string, ProductInStore> keyValue in Shop_Manager.StoresList[Sber_ID].AllProductInStore)
            {
                Console.WriteLine(keyValue.Key + " Product name: " + keyValue.Value.productName + " -- Quantity:  " +
keyValue.Value.quantity + " Pricetag: " + keyValue.Value.price);
            }
            Console.WriteLine("List of products in the Yandex Lavka");
            foreach (KeyValuePair<string, ProductInStore> keyValue in Shop_Manager.StoresList[Yandex_ID].AllProductInStore)
            {
                Console.WriteLine(keyValue.Key + " Product name: " + keyValue.Value.productName + " -- Quantity:  " +
keyValue.Value.quantity + " Pricetag: " + keyValue.Value.price);
            }
            Console.WriteLine("List of products in the Utkonos");
            foreach (KeyValuePair<string, ProductInStore> keyValue in Shop_Manager.StoresList[Utkonos_ID].AllProductInStore)
            {
                Console.WriteLine(keyValue.Key + " Product name: " + keyValue.Value.productName + " -- Quantity:  " +
keyValue.Value.quantity + " Pricetag: " + keyValue.Value.price);
            }

            List<KeyValuePair<ProductInStore, int>> ProductsAvailableListInSberForBuy = new List<KeyValuePair<ProductInStore, int>>();
            ProductsAvailableListInSberForBuy = Shop_Manager.StoresList[Sber_ID].available_product(1000);
            foreach (KeyValuePair<ProductInStore, int> keyValue in ProductsAvailableListInSberForBuy)
            {
                Console.WriteLine("You can buy in Sber Market for 1000 rubles: " + keyValue.Value + " of " + keyValue.Key.productName);
            }
            List<KeyValuePair<ProductInStore, int>> ProductsAvailableListInYandexForBuy = new List<KeyValuePair<ProductInStore,
int>>();
            ProductsAvailableListInYandexForBuy = Shop_Manager.StoresList[Yandex_ID].available_product(1000);
            foreach (KeyValuePair<ProductInStore, int> keyValue in ProductsAvailableListInYandexForBuy)
            {
                Console.WriteLine("You can buy in Yandex Lavka for 1000 rubles: " + keyValue.Value + " of " + keyValue.Key.productName);
            }
            List<KeyValuePair<ProductInStore, int>> ProductsAvailableListInUtkonosForBuy = new List<KeyValuePair<ProductInStore,
int>>();
            ProductsAvailableListInUtkonosForBuy = Shop_Manager.StoresList[Utkonos_ID].available_product(1000);
            foreach (KeyValuePair<ProductInStore, int> keyValue in ProductsAvailableListInUtkonosForBuy)
            {
                Console.WriteLine("You can buy in Utkonos for 1000 rubles: " + keyValue.Value + " of " + keyValue.Key.productName);
            }

            Dictionary<string, int> MyCartForSber = new Dictionary<string, int>
                                { {tea_ID, 1}, {bread_ID, 2}, {milk_ID, 1}, {lemon_ID, 2}, {water_ID, 5} };
            Dictionary<string, int> MyCartForYandex = new Dictionary<string, int>
                                { {tea_ID, 1}, {bread_ID, 2}, {milk_ID, 1}, {lemon_ID, 2}, {water_ID, 5} };
            Dictionary<string, int> MyCartForUtkonos = new Dictionary<string, int>
                                { {tea_ID, 1}, {bread_ID, 2}, {milk_ID, 1}, {lemon_ID, 2}, {water_ID, 5} };

            double amountCostSellSber = Shop_Manager.StoresList[Sber_ID].get_sell_amount_cost(MyCartForSber);
            double amountCostSellYandex = Shop_Manager.StoresList[Yandex_ID].get_sell_amount_cost(MyCartForYandex);
            double amountCostSellUtkonos = Shop_Manager.StoresList[Utkonos_ID].get_sell_amount_cost(MyCartForUtkonos);

            Console.WriteLine("Amount cost in Sber Market " + amountCostSellSber);
            Console.WriteLine("Amount cost in Yandex Lavka " + amountCostSellYandex);
```

```
            Console.WriteLine("Amount cost in Utkonos " + amountCostSellUtkonos);

            Console.WriteLine("Better to buy in " + Shop_Manager.best_store(MyCartForSber).storeName);



        }
        catch (Store_Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
        finally
        {
            stopWatch.Stop();
            TimeSpan ts = stopWatch.Elapsed;
            string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
                ts.Hours, ts.Minutes, ts.Seconds,
                ts.Milliseconds / 10);
            Console.WriteLine("RunTime " + elapsedTime);
        }

    }
  }
}
```

Файл ProductInStore.cs

```
using System;
namespace Code_of_lab_2
{
    public class ProductInStore : Product
    {
        public int    quantity { get; set; }
        public double price    { get; set; }

        public ProductInStore(Product curentProduct, int quantity, double price)
            : base(curentProduct.productName)
        {
            productID = curentProduct.productID;
            this.quantity = quantity;
            this.price = price;
        }
    }
}
```