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 М3201:

Дымчикова Аюна

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

В результате выполнения работы была реализована система Product (товар) -> Shipment (партия товаров) -> Shop (магазин) -> ShopNet (сеть магазинов).

Класс товара обладает только полями названия и артикула.

Партия товаров состоит из следующих полей: товар, цена за единицу товара и количество товара. При этом для партии товаров, в отличие от класса самого товара, существует возможность изменения полей цены и количества.

Магазин состоит из полей с именем, артикулом и коллекцией партий товара. Данный класс необходим для хранения информации о партиях товара в данном магазине, при этом в одном магазине могут существовать партии товара с одинаковым названием, если их артикул отличен друг от друга. Данный класс обладает методами для изменения полей партии товаров, а также методом для подсчета количества товаров, которое можно купить на заданную сумму в данном экземпляре класса.

Сеть магазинов состоит из объединения магазинов. Данный класс используется для поиска магазина с наиболее дешевым определенным продуктом или списком продуктов.

Таким образом, были реализованы методы для всех операций, представленных в условии лабораторной работы, а также реализована система классов для удобной работы с данными.

Листинг

Файл Main.java

```
import java.util.ArrayList;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        try {
            //filling the shops
            Product chocolate = new Product("Chocolate", 1);
            Product melon = new Product("Melon", 2);
            Product bread = new Product("White bread", 3);
            Product butter = new Product("Butter", 4);
            Product cereal = new Product("Cereal", 5);
            Product apple = new Product("Apple", 6);
            Product orange = new Product("Orange", 7);
            Product table = new Product("table", 8);
            Product cup = new Product("Cup", 9);
            Product milk = new Product("Milk", 10);
            Product sugar = new Product("White sugar", 11);
            Product flour = new Product("Flour", 12);
            Product flour1 = new Product("Flour", 13);



            Shop A = new Shop("Shop A", 1);
            Shop B = new Shop("Shop B", 2);
            Shop C = new Shop("Shop C", 3);

            A.add_shipment(chocolate, 10, 5);
            B.add_shipment(chocolate, 50, 10);
            C.add_shipment(chocolate, 100, 100);

            A.add_shipment(melon, 100, 15);
            B.add_shipment(melon, 150, 20);

            A.add_shipment(bread, 70, 100);
            B.add_shipment(bread, 80, 20);

            C.add_shipment(butter, 98, 50);
            A.add_shipment(cereal, 100, 30);
            B.add_shipment(apple, 80, 90);
            C.add_shipment(orange, 150, 25);
            A.add_shipment(table, 100000, 100000);
            B.add_shipment(cup, 50, 15);
            C.add_shipment(milk, 67, 250);

            ShopNet net = new ShopNet(A, B, C);
            //try to get the number of shop with the cheapest chocolate
            System.out.println(net.get_shop_name(net.get_cheapest_product(chocolate)) +
```

```
        " : " + chocolate.get_name() +
        " : " + net.get_shop(net.get_cheapest_product(chocolate)).get_price(chocolate));

//try to find the cheapest list with right number of products
//choco: A - 5; B - 10; C - 100
ArrayList<Map.Entry<Product, Integer>> product_list = new ArrayList<>();
product_list.add(Map.entry(chocolate, 10));
product_list.add(Map.entry(melon, 1));
System.out.println(net.get_shop_name(net.get_cheapest_list(product_list)) +
        " : " + net.get_shop(net.get_cheapest_list(product_list)).try_shipment(
        product_list.get(0).getKey(), product_list.get(0).getValue()).getValue());

//try to find the cheapest list in the right shop
//all products are the most cheapest in the shop A
ArrayList<Map.Entry<Product, Integer>> product_list1 = new ArrayList<>();
product_list1.add(Map.entry(chocolate, 5));
product_list1.add(Map.entry(melon, 1));
product_list1.add(Map.entry(bread, 20));
System.out.println(net.get_shop_name(net.get_cheapest_list(product_list1)));

//is all right with int/long
product_list1.add(Map.entry(table, 100000));
System.out.println(net.get_shop_name(net.get_cheapest_list(product_list1)));

//is all right with set price
Shop D = new Shop("Shop D", 4);
D.add_shipment(flour, 1, 15);
System.out.println(D.get_price(flour));
D.set_price(flour, 10);
System.out.println(D.get_price(flour));

//is all right with different products with the same name
D.add_shipment(flour1, 100, 100);
System.out.println(D.get_price(flour1));
System.out.println(D.get_price(flour));

//get the list of products that cost less or equal to the limit
ArrayList<Map.Entry<Product, Integer>> list = A.get_limit_buy(100);
for(Map.Entry<Product, Integer> entry : list) {
   System.out.print(entry.getKey().get_name() + " : " + entry.getValue() + "|");
}

//price = 0
D.add_shipment(sugar, 0, 10);
System.out.println("\n" + D.try_shipment(sugar, 10).getKey() + " " + D.try_shipment(sugar, 10).getValue());
System.out.println(D.try_shipment(sugar, 100).getKey());

//exceptions
//D.set_price(sugar, -10);
//D.add_shipment(sugar, -15, 1);
//D.add_shipment(sugar, 15, -1);
//D.set_price(apple, 5);
```

```
            //D.set_number(apple, 3);
            //System.out.println(net.get_cheapest_product(sugar));
            //product_list1.add(Map.entry(sugar, 5));
            //System.out.println(net.get_cheapest_list(product_list1));

        } catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Файл Product.java

```
public class Product {
    private String name = "";
    private int code = 0;

    public Product() {}
    public Product(String _name, int _code) {
        name = _name;
        code = _code;
    }
    public Product(Product other) {
        this(other.get_name(), other.get_code());
    }

    public String get_name() {
        return name;
    }
    public int get_code() {
        return code;
    }
}
```

Файл Shipment.java

```
public class Shipment {
    private Product product;
    private int price = 0;
    private int number = 0;

    public Shipment() {};
    public Shipment(Product _product, int _price, int _number) throws Exception {
        product = _product;
        set_price(_price);
        set_number(_number);
    }

    public int get_number() {
        return number;
    }
    public int get_price() {
        return price;
    }
    public Product get_product() {
        return product;
    }
    public void set_price(int _price) throws Exception {
        if(_price < 0) {
            throw new Exception("Shipment can't have a negative price of products!");
        }
        price = _price;
    }
    public void set_number(int _number) throws Exception {
        if(_number >= 0) {
            number = _number;
        } else if(number >= -_number) {
            number += _number;
        } else {
            throw new Exception("Not enough products of type: " + product.get_name());
        }
    }
}
```

Файл Shop.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


public class Shop {
    private String name = "";
    private int code = 0;
    private Map<Integer, Shipment> assortiment = new HashMap<>();

    public Shop() {};
    public Shop(String _name, int _code) {
        name = _name;
        code = _code;
    }

    public void add_shipment(Product product, int price, int number) throws Exception {
        if(number == 0) {
            return;
        }
        if(number < 0) {
            throw new Exception("Shipment can't have a negative number of products!");
        }
        Shipment shipment = assortiment.getOrDefault(product.get_code(), new Shipment());
        assortiment.put(product.get_code(), new Shipment(product, price, shipment.get_number() + number));
    }
    public void set_price(Product product, int price) throws Exception {
        if(assortiment.containsKey(product.get_code())) {
            assortiment.get(product.get_code()).set_price(price);
        } else {
            throw new Exception("No such product: " + product.get_name());
        }
    }
    public int get_price(Product product) throws Exception {
        if(assortiment.containsKey(product.get_code())) {
            return assortiment.get(product.get_code()).get_price();
        } else {
            throw new Exception("No such product: " + product.get_name());
        }
    }
    public boolean has_product(Product product) {
        if(assortiment.containsKey(product.get_code())) {
            return true;
        }
        return false;
    }
    private void set_number(Product product, int number) throws Exception {
        if(assortiment.containsKey(product.get_code())) {
```

```
        if(number == 0) {
            assortiment.remove(product.get_code());
        } else {
            assortiment.get(product.get_code()).set_number(number);
        }
    } else {
        throw new Exception("There is no such product: " + product.get_name());
    }
}

public Map.Entry<Boolean, Long> try_shipment(Product product, int number) {
    if(number == 0) {
        return Map.entry(true, 0L);
    }
    if(number < 0 || assortiment.getOrDefault(product.get_code(), new Shipment()).get_number() < number) {
        return Map.entry(false, 0L);
    }
    return Map.entry(true, (long) (assortiment.get(product.get_code()).get_price() * number));
}

public void buy_shipment(Product product, int number) throws Exception {
    if(number < 0) {
        throw new Exception("Was an attempt to buy a negative number of products");
    }
    set_number(product, -number);
}
public ArrayList<Map.Entry<Product, Integer>> get_limit_buy(int limit) {
    ArrayList<Map.Entry<Product, Integer>> res = new ArrayList<>();
    for (Map.Entry<Integer, Shipment> entry : assortiment.entrySet()) {
        int price = entry.getValue().get_price();
        if(price > 0) {
                //&& limit / price > 0 && (limit / price) <= entry.getValue().get_number()) {
            //res.add(Map.entry(entry.getValue().get_product(), limit / price));
            res.add(Map.entry(entry.getValue().get_product(), Integer.min(limit / price, entry.getValue().get_number())));
        } else if (price == 0) {
            res.add(Map.entry(entry.getValue().get_product(), entry.getValue().get_number()));
        }
    }
    return res;
}
public String get_name() {
    return name;
}
public int get_code() {
    return code;
}

public String get_name(Product product) {
    return assortiment.get(product.get_code()).get_product().get_name();
}
public int get_number(Product product) {
    return assortiment.get(product.get_code()).get_number();
```

```
    }
}
```

Файл ShopNet.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class ShopNet {
    private Map<Integer, Shop> net = new HashMap<>();

    public ShopNet() {};
    public ShopNet(Shop shop) {
        net.put(shop.get_code(), shop);
    }
    public ShopNet(Shop ... shop) {
        for(int i = 0; i < shop.length; i++) {
            net.put(shop[i].get_code(), shop[i]);
        }
    }

    public void add_shop(Shop shop) {
        net.put(shop.get_code(), shop);
    }
    public Shop get_shop(int code) {
        return net.get(code);
    }
    public int get_cheapest_product(Product product) throws Exception {
        int code = -1;
        int price = Integer.MAX_VALUE;
        for (Map.Entry<Integer, Shop> entry : net.entrySet()) {
            if(entry.getValue().has_product(product) && entry.getValue().get_price(product) < price) {
                price = entry.getValue().get_price(product);
                code = entry.getValue().get_code();
            }
        }
        if(code == -1) {
            throw new Exception("No such product: " + product.get_name());
        }
        return code;
    }

    public int get_cheapest_list(ArrayList<Map.Entry<Product, Integer>> product_list) throws Exception {
        long sum = Long.MAX_VALUE;
        int code = -1;
        for (Map.Entry<Integer, Shop> shop_entry : net.entrySet()) {
            long cost = 0;
            for(Map.Entry<Product, Integer> shipment : product_list) {
                Map.Entry<Boolean, Long> try_cost = shop_entry.getValue().try_shipment(shipment.getKey(), shipment.getValue());
                int c = 8;
                if(!try_cost.getKey()) {
                    cost = Long.MAX_VALUE;
```

```
                break;
            }
            cost += try_cost.getValue();
        }
        if (cost < sum) {
            sum = cost;
            code = shop_entry.getValue().get_code();
        }
    }
    if(code == -1) {
        throw new Exception("No such product list in one shop");
    }
    return code;
}

public String get_shop_name(int code) {
    return get_shop(code).get_name();
}
}
```