МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №5

по дисциплине: ООП
тема: «Классы, виды отношений. Наследование.»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
пр. Черников Сергей Викторович

Белгород 2024 г.

Лабораторная работа №5

«Классы»

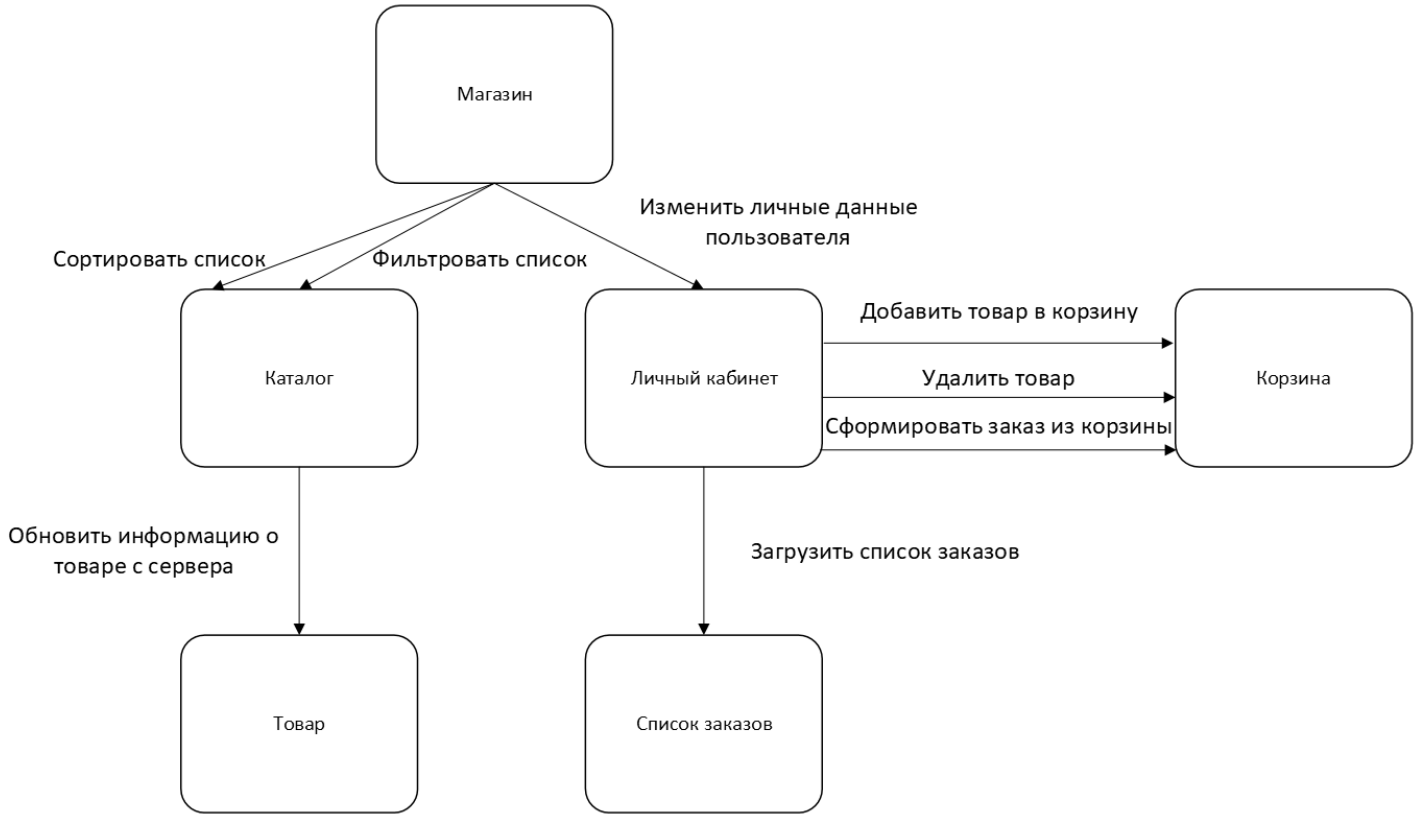
Вариант 6

Цель работы: получение теоретических знаний в области разработки классов, получение практических навыков реализаций классов и отношений между ними.

Задание 1

Выполнить построение объектной модели заданной предметной области:

Система закупок.



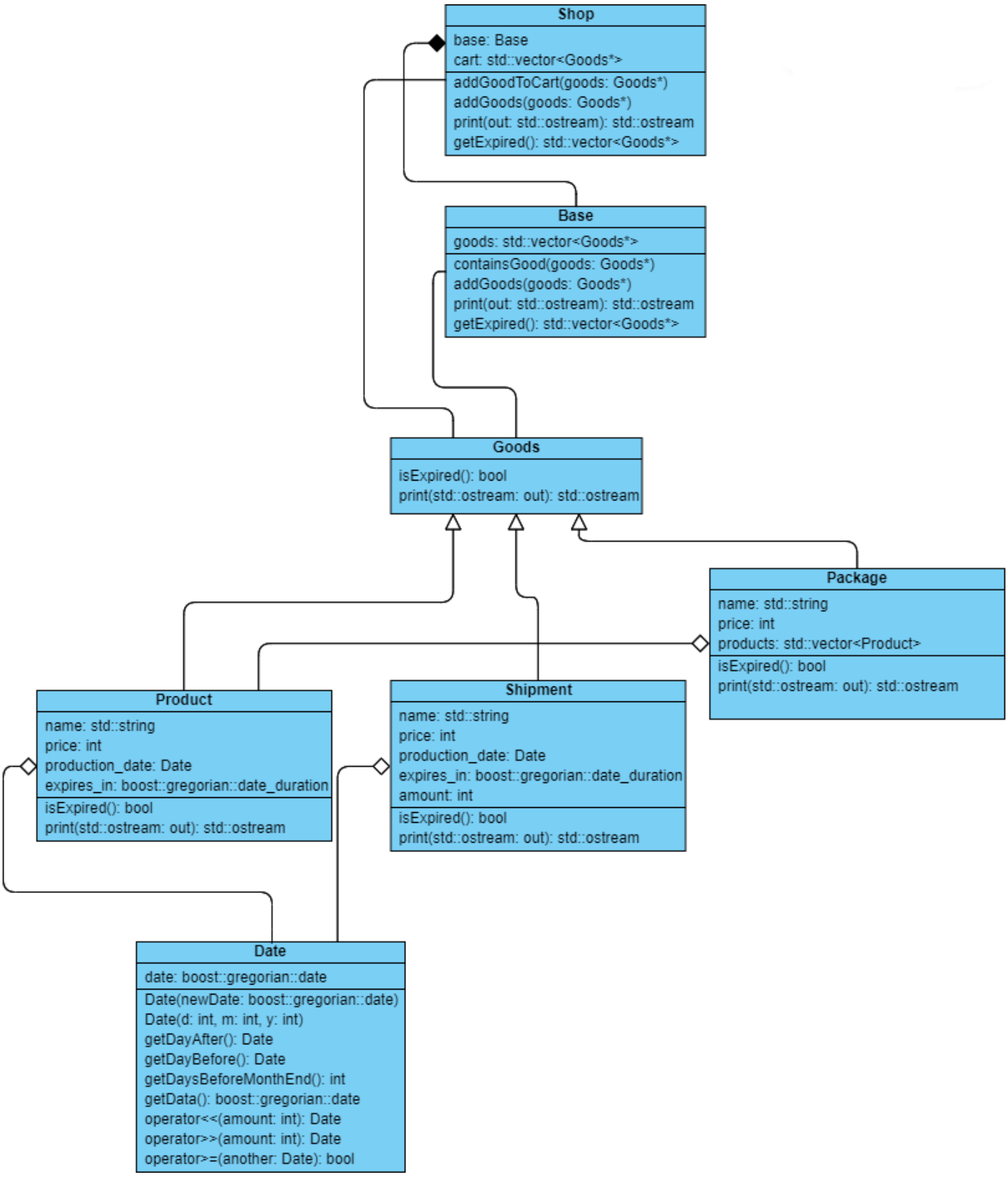
Задание 2

Разработать диаграмму классов для описанной объектной модели, и реализовать предложенные классы.

1. Создать абстрактный класс Товар с методами, позволяющим вывести на экран информацию о товаре, а также определить, соответствует ли она сроку годности на текущую дату.

2. Создать производные классы: Продукт (название, цена, дата производства, срок годности), Партия (название, цена, количество шт., дата производства, срок годности), Комплект (названия, цена, перечень продуктов) со своими методами вывода информации на экран, и определения соответствия сроку годности.

3. Создать базу (массив) из n товаров, вывести полную информацию из базы на экран, а также организовать поиск просроченного товара (на момент текущей даты).



goods.h

```cpp
#pragma once

#include "../date/date.h"

class Goods {
public:
    virtual bool isExpired() = 0;

    virtual std::ostream& print(std::ostream& out) = 0;
```

```
};

class Product : public Goods {
    std::string name;
    int price;
    Date production_date;
    boost::gregorian::date_duration expires_in;
public:
    Product(std::string name,
            int price,
            Date production_date,
            boost::gregorian::date_duration expirates_in) :
        name(name), price(price), production_date(production_date), expires_in(expirates_in) {};

    virtual bool isExpired() override;
    virtual std::ostream& print(std::ostream& out) override;
};

class Shipment : public Goods {
    std::string name;
    int price;
    int amount;
    Date production_date;
    boost::gregorian::date_duration expires_in;
public:
    Shipment(std::string name,
            int price,
            int amount,
            Date production_date,
            boost::gregorian::date_duration expirates_in) :
        name(name), price(price), amount(amount), production_date(production_date), expires_in(expirates_in) {};

    virtual bool isExpired() override;
    virtual std::ostream& print(std::ostream& out) override;
};

class Package : public Goods {
    std::string name;
    int price;
    std::vector<Product> products;
public:
    Package(std::string name,
             int price,
             std::vector<Product> products) :
        name(name), price(price), products(products) {};

    virtual bool isExpired() override;
    virtual std::ostream& print(std::ostream& out) override;
};

class Base {
    std::vector<Goods*> goods;
public:
    bool containsGoods(Goods* goods);
```

```
    void addGoods(Goods* goods);
    std::ostream& print(std::ostream& out);
    std::vector<Goods*> getExpired();
};

class Shop {
    Base base;
    std::vector<Goods*> cart;
public:
    void addGoods(Goods* goods);
    void addGoodToCart(Goods* good);
    std::ostream& print(std::ostream& out);
    std::vector<Goods*> getExpired();
};
```

goods.cpp

```
#include "goods.h"

bool Product::isExpired() {
    boost::gregorian::date current_date(boost::gregorian::day_clock::local_day());
    Date now(current_date);
    Date exDate = Date(this->production_date.getData() + this->expires_in);
    return now >= exDate;
}

std::ostream& Product::print(std::ostream& out) {
    out << "Name: " << this->name << "\n";
    out << "Price: " << this->price << "\n";
    out << "Production date: " << this->production_date.getData() << "\n";
    out << "Expires at: " << this->production_date.getData() + this->expires_in << "\n";
    out << "Expired: " << (this->isExpired() ? "yup" : "nuh uh") << "\n";

    return out;
}

bool Shipment::isExpired() {
    boost::gregorian::date current_date(boost::gregorian::day_clock::local_day());
    Date now(current_date);
    Date exDate = Date(this->production_date.getData() + this->expires_in);
    return now >= exDate;
}

std::ostream& Shipment::print(std::ostream& out) {
    out << "Name: " << this->name << "\n";
    out << "Price: " << this->price << "\n";
    out << "Amount: " << this->amount << "\n";
    out << "Production date: " << this->production_date.getData() << "\n";
    out << "Expires at: " << this->production_date.getData() + this->expires_in << "\n";
    out << "Expired: " << (this->isExpired() ? "yup" : "nuh uh") << "\n";

    return out;
}
```

```cpp
bool Package::isExpired() {
    for (auto& good : this->products)
        if (good.isExpired()) return true;

    return false;
}

std::ostream& Package::print(std::ostream& out) {
    out << "Name: " << this->name << "\n";
    out << "Price: " << this->price << "\n";
    out << "Products:\n";
    out << "=============\n";
    for (auto& prod : this->products) {
        prod.print(out);
    }
    out << "=============\n";

    return out;
}

bool Base::containsGoods(Goods* goods) {
    return std::find(this->goods.begin(), this->goods.end(), goods) != this->goods.end();
}

void Base::addGoods(Goods* goods) {
    if (!containsGoods(goods))
        this->goods.push_back(goods);
}

std::ostream& Base::print(std::ostream& out) {
    std::cout << "*************\n";

    for (auto & good : this->goods) {
        good->print(out);
        std::cout << "*************\n";
    }

    return out;
}

std::vector<Goods*> Base::getExpired() {
    std::vector<Goods*> result;
    for (auto & good : this->goods)
        if (good->isExpired())
            result.push_back(good);

    return result;
}

void Shop::addGoods(Goods* goods) {
    this->base.addGoods(goods);
}
```

```
void Shop::addGoodToCart(Goods* good) {
    if (this->base.containsGoods(good))
        this->cart.push_back(good);
}

std::ostream& Shop::print(std::ostream& out) {
    std::cout << "Catalog: \n";
    this->base.print(std::cout);
    std::cout << "Cart: \n";

    std::cout << "*************\n";

    for (auto & good : this->cart) {
        good->print(out);
        std::cout << "*************\n";
    }

    return out;
}

std::vector<Goods*> Shop::getExpired() {
    return this->base.getExpired();
}
```

main.cpp

```
#include <iostream>
#include <windows.h>

#include "../libs/alg/goods/goods.h"

int main() {
    Shop shop;
    shop.addGoods(new Product("Monitor", 10000, Date(5, 3, 2024), boost::gregorian::days(3000)));
    shop.addGoods(new Shipment("Sneakers", 3000, 150, Date(3, 3, 2023), boost::gregorian::days(3000)));
    shop.addGoods(new Package("Vegetrables", 10313,
                              {Product("Tomatoes", 100, Date(5, 3, 2024), boost::gregorian::days(45)),
                               Product("Cucumber", 100, Date(5, 3, 2024), boost::gregorian::days(45)),
                               Product("Salad", 300, Date(5, 3, 2024), boost::gregorian::days(45))}));
    shop.addGoods(new Product("Tomatoes", 100, Date(5, 3, 2024), boost::gregorian::days(0)));

    shop.print(std::cout);
}
```

Результаты выполнения программы:

```
Catalog:
*************
Name: Monitor
Price: 10000
Production date: 2024-Mar-05
Expires at: 2032-May-22
Expired: nuh uh
```

```
*************
Name: Sneakers
Price: 3000
Amount: 150
Production date: 2023-Mar-03
Expires at: 2031-May-20
Expired: nuh uh
*************
Name: Vegetrables
Price: 10313
Products:
=============
Name: Tomatoes
Price: 100
Production date: 2024-Mar-05
Expires at: 2024-Apr-19
Expired: nuh uh
Name: Cucumber
Price: 100
Production date: 2024-Mar-05
Expires at: 2024-Apr-19
Expired: nuh uh
Name: Salad
Price: 300
Production date: 2024-Mar-05
Expires at: 2024-Apr-19
Expired: nuh uh
=============
*************
Name: Tomatoes
Price: 100
Production date: 2024-Mar-05
Expires at: 2024-Mar-05
Expired: yup
*************
Cart:
*************
```

Ссылка на репозиторий

Вывод: в ходе лабораторной работы получены теоретические знания в области разработки классов, получены практические навыки реализаций классов и отношений между ними.