**Что нужно сделать**

Ваша задача — **создать структуру классов для интернет-магазина**. Над этой структурой вы будете работать еще в нескольких домашках, поэтому сохраните этот проект.

* Создайте обычное Java-приложение с одним классом

App

 и методом

main()

.

* Сделайте коммит в ветку main(master).
* После этого создайте новую ветку и выполните в ней домашнее задание.

**Описание структуры**

Пока наш интернет-магазин будет содержать только классы товаров и корзины для товаров. Эти классы нужно создать с учетом принципа инкапсуляции и разнести по пакетам.

Выполняйте задание строго по шагам. Пока не сделали предыдущий шаг — не приступайте к следующему.

**Создание корневого пакета**

1. Создайте в папке

src

 пакет с названием

org.skypro.skyshop

. Этот пакет будет корневым для всех остальных пакетов в нашем приложении.

1. Перенесите в этот пакет класс

App

 с методом

main

. Его нужно обязательно перенести, а не копировать. Для переноса можно использовать Drag and Drop или комбинацию клавиш *«Вырезать»/«Вставить»*.

Для последующих домашних заданий этого курса вам нужно создавать корневой пакет, иначе работа не будет принята.

**Создание класса товара**

Создайте в пакете

org.skypro.skyshop.product

 класс товара

Product

, который будет содержать два значения:

* название продукта в виде строки,
* стоимость продукта в виде целого числа.

Эти значения должны быть немодифицируемыми после создания объекта, но должна быть возможность получить эти значения в других классах.

Подсказка

Если значения должны быть немодифицируемыми — это значит, что в нашем классе должен быть единственный конструктор, принимающий эти значения. Сами поля должны иметь модификатор

private

, а доступ к ним должен осуществляться с помощью геттеров.

**Создание класса корзины для товаров**

Создайте простой класс корзины

ProductBasket

 в пакете

org.skypro.skyshop.basket

.

Этот класс содержит в себе одну корзину с продуктами, которые купил один пользователь. Вам нужно реализовать следующие публичные методы:

1. Метод добавления продукта в корзину: метод принимает в себя продукт и ничего не возвращает.
2. Метод получения общей стоимости корзины: метод ничего не принимает и возвращает целое число.
3. Метод, который печатает содержимое корзины: метод ничего не принимает и не возвращает, но печатает в консоль сообщение вида:
4. <имя продукта>: <стоимость>
5. <имя продукта>: <стоимость>
6. <имя продукта>: <стоимость>

Итого: <общая стоимость корзины>

Если в корзине ничего нет, нужно напечатать фразу *«в корзине пусто»*.

1. Метод, проверяющий продукт в корзине по имени: метод принимает в себя строку имени и возвращает

boolean

 в зависимости от того, есть продукт в корзине или его нет.

1. Метод очистки корзины: метод ничего не принимает и очищает массив, проставляя всем его элементам

null

.

**Обратите внимание!**

1. В качестве хранилища для объектов product используйте массив из пяти элементов, однако прямой доступ к этому массиву должен быть невозможен.
2. Если в массиве не хватает места на следующий продукт, то метод добавления продукта должен выводить в консоль сообщение вида: «Невозможно добавить продукт».

**Демонстрация классов**

Создайте в классе

App

 в методе

main

 несколько продуктов и продемонстрируйте работу всех методов корзины по следующему сценарию:

1. Добавление продукта в корзину.
2. Добавление продукта в заполненную корзину, в которой нет свободного места.
3. Печать содержимого корзины с несколькими товарами.
4. Получение стоимости корзины с несколькими товарами.
5. Поиск товара, который есть в корзине.
6. Поиск товара, которого нет в корзине.
7. Очистка корзины.
8. Печать содержимого пустой корзины.
9. Получение стоимости пустой корзины.
10. Поиск товара по имени в пустой корзине.

**Рекомендации по выполнению работы**

1. Используйте модификатор

static

 только в методе

main

, а модификатор

public

— только для методов и конструкторов.

1. Для демонстрации классов можно создать несколько корзин.