**Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации**

**Ордена Трудового Красного Знамени Федеральное**

**государственное бюджетное образовательное**

**учреждение высшего образования**

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**СВЯЗИ И ИНФОРМАТИКИ**

**Лабораторная работа № 3**

по дисциплине: «Рефакторинг баз данных и приложений»

**по теме:**

**«Изучение рефакторинга приложений»**

Выполнил: студент группы БПИ-2303

Гавриленко В. А.

Проверил:

Москва, 2024

**Цель работы:**

1. Ознакомиться с основными принципами и задачами рефакторинга.
2. Научиться выявлять проблемные участки кода (code smells) и устранять их.
3. Применить техники рефакторинга для улучшения читаемости, структуры и производительности кода.
4. Развить навыки анализа и улучшения существующего кода.

**Задание:**

1. Ознакомьтесь с теоретическими основами рефакторинга, включая его цели, преимущества и отличия от оптимизации. Изучите основные техники рефакторинга, такие как: разделение больших функций на более мелкие, устранение дублирующегося кода, улучшение именования переменных, функций и классов, введение уровней абстракции.
2. Проведите анализ предоставленного кода, чтобы выявить проблемные участки. Составьте список проблемных мест, которые требуют рефакторинга.
3. Примените техники рефакторинга для устранения выявленных проблем.
4. Подготовьте отчет, включающий описание исходного состояния кода (с примерами проблемных участков), внесенные изменения (с пояснениями и обоснованием), итоговое состояние кода.
5. Проверьте, что после рефакторинга программа работает корректно. Если возможно, напишите или обновите тесты для измененных участков кода.ы

**Ход работы:**

Для начала, я нашел код, который идеально подходит для рефакторинга. Это небольшая программа на языке программирования Python, которая представляет собой упрощенную систему оформления заказов в магазине. Она позволяет добавлять заказы (переменные: товар, цена, количество), считать общую стоимость всех заказов, а также печатать чек с перечнем товаров и итоговой суммой покупки.

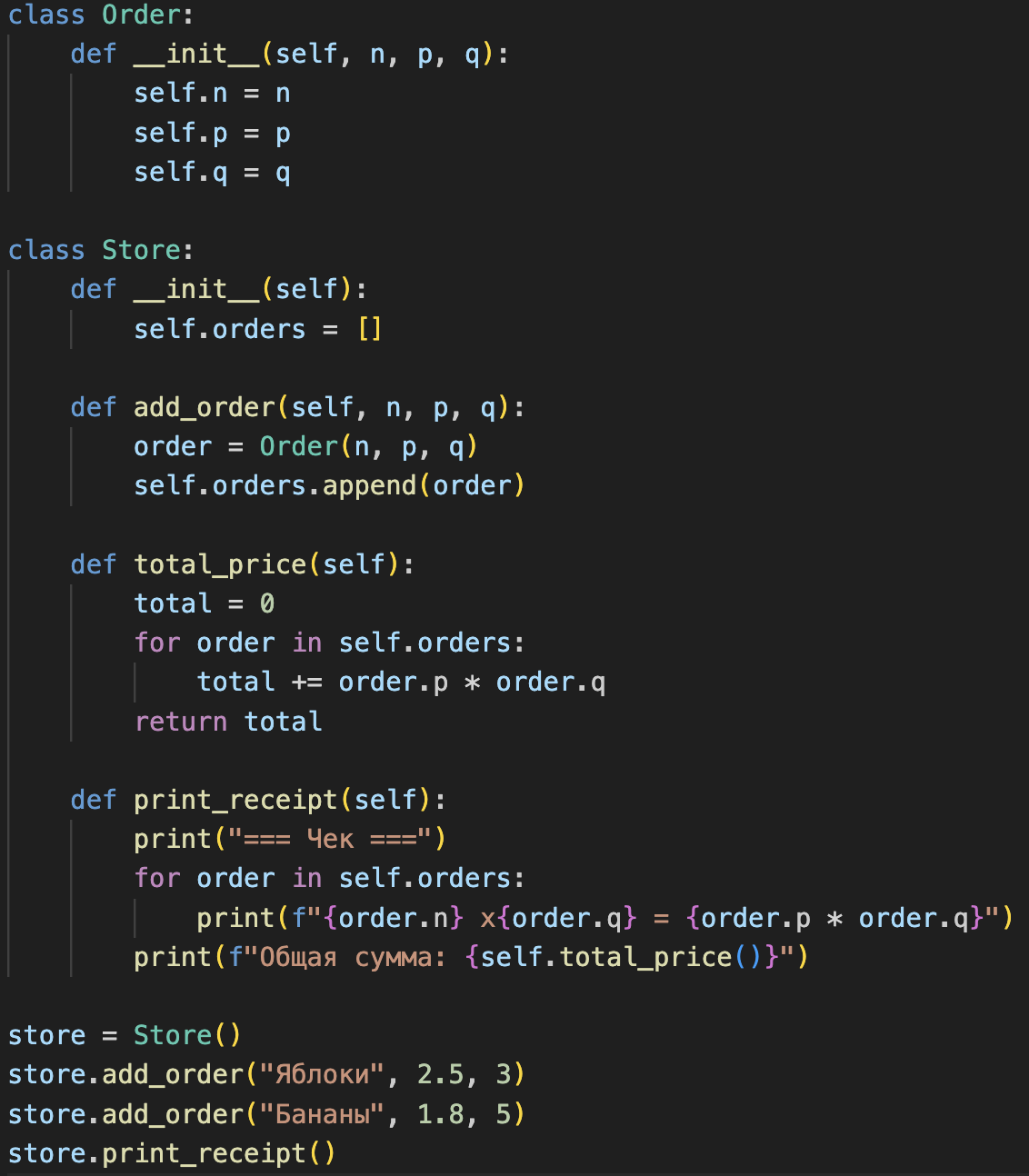


Рис. 1. Изначальный код программы до рефакторинга.

В коде содержатся следующие проблемы:

1. плохие имена переменных (n, p, q), которые непонятно, что означают дублирование кода (расчет заказов встречается дважды)
2. нарушение принципов ООП (Store хранит заказы и печатает чек,

переменные класса Order (n, p, q) можно менять напрямую)

1. отсутствие разделения логики (расчет общей суммы и вывод чека смешаны в одном классе)

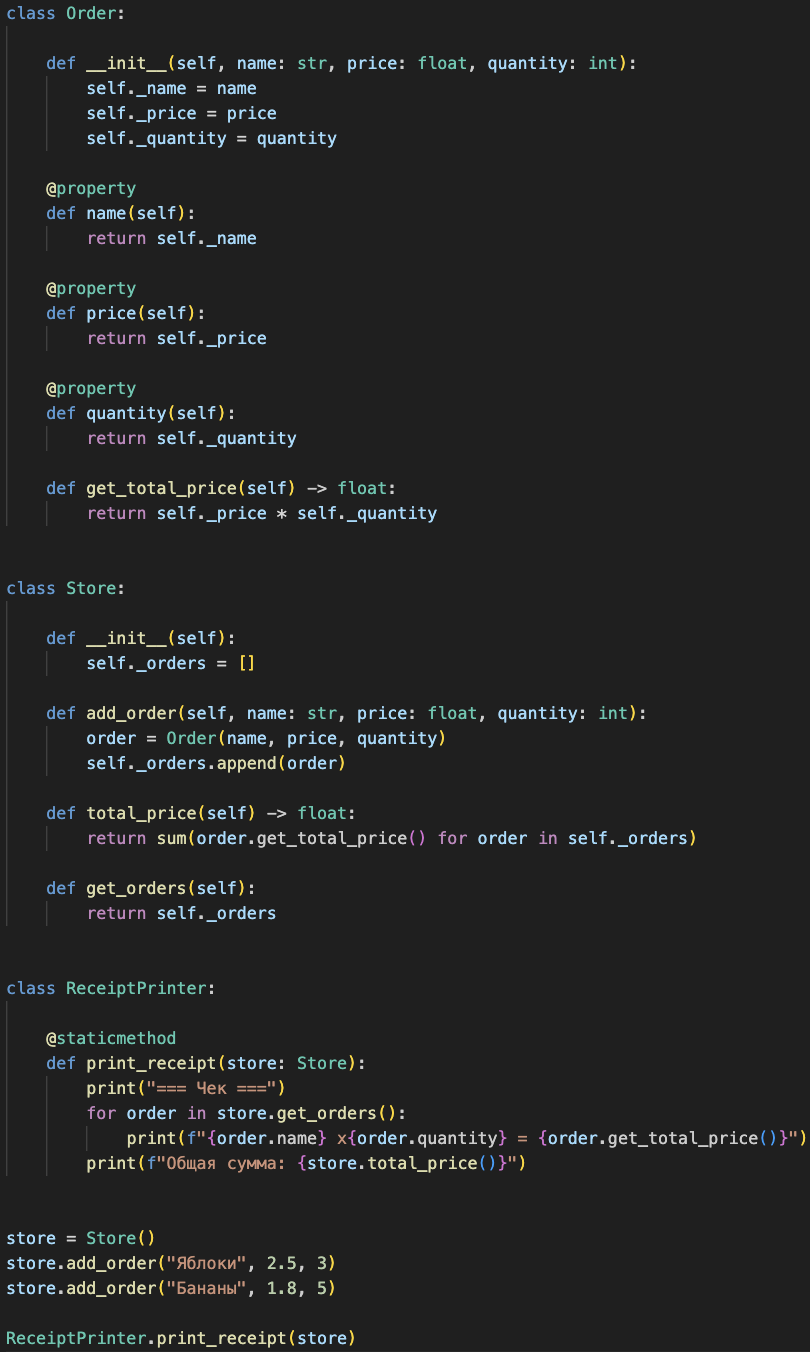


Рис. 2. Код после рефакторинга

Что было сделано:

1. Улучшены имена переменных (n стал name, p стал price, q стал quantity)

2. Избежали дублирования кода (добавлен метод get\_total\_price() )

3. Код написан с принципом ООП (Store теперь только хранит заказы, ReceiptPrinter теперь только печатает чек, поля Order стали приватными (\_name, \_price, \_quantity))

4. Использовали sum() (замена for-цикла, делая код более читаемым)

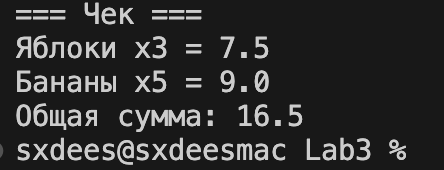


Рис. 3. Проверка корректной работы кода после рефакторинга

**Вывод:**

В ходе выполнения лабораторной работы я ознакомился с принципами рефакторинга кода. Затем на практике был обработан код, у которого произошли улучшения читаемости, структуры и производительности.