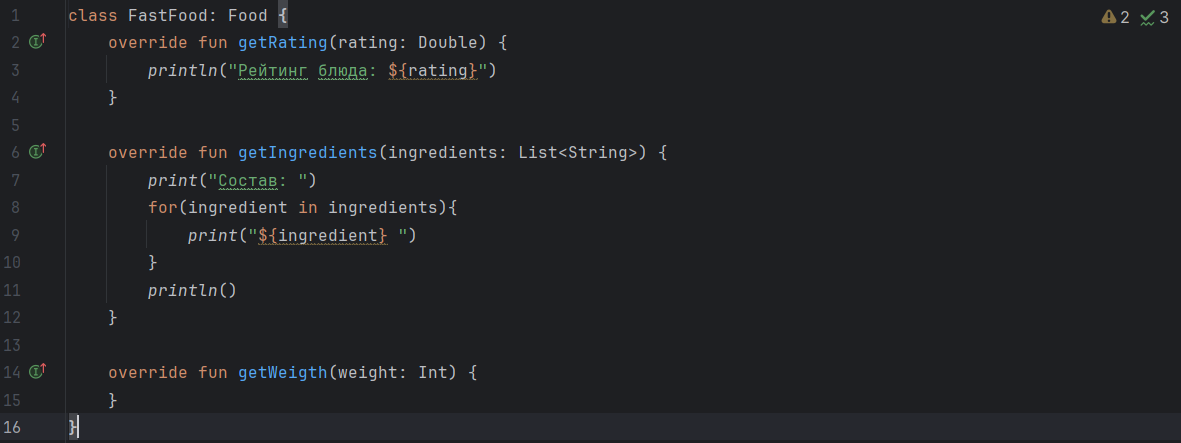
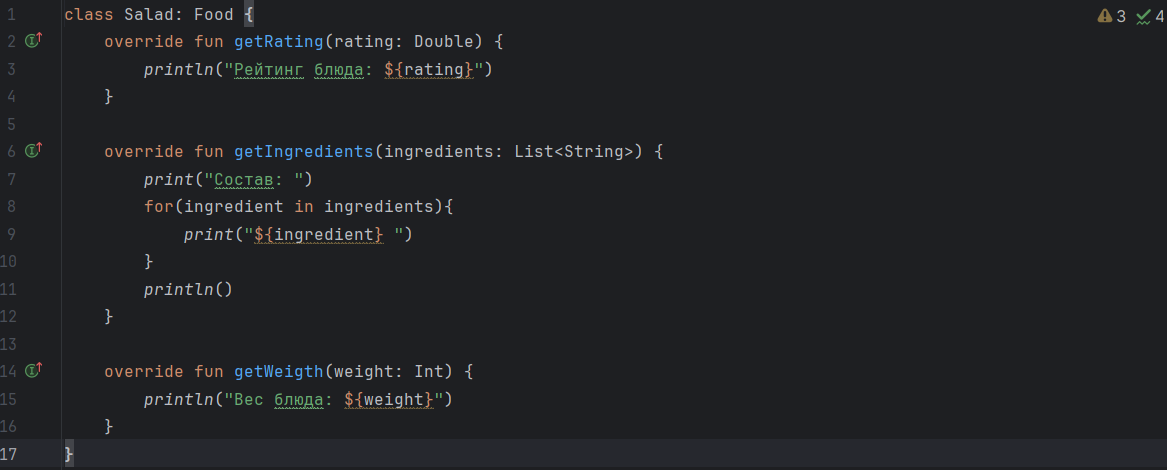
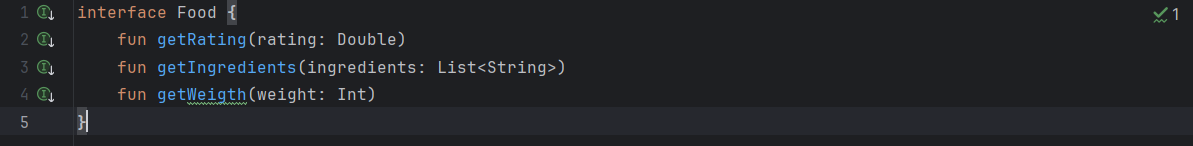
**Паттерн Посетитель(Visitor)**

**Версия без паттерна**

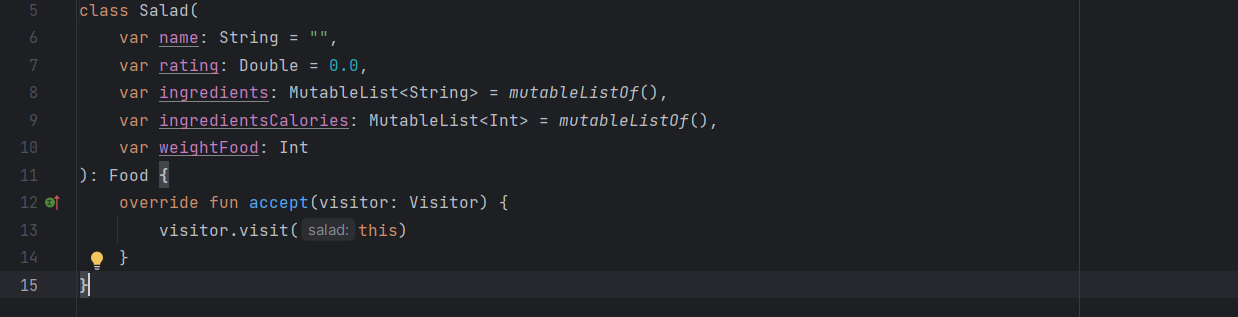
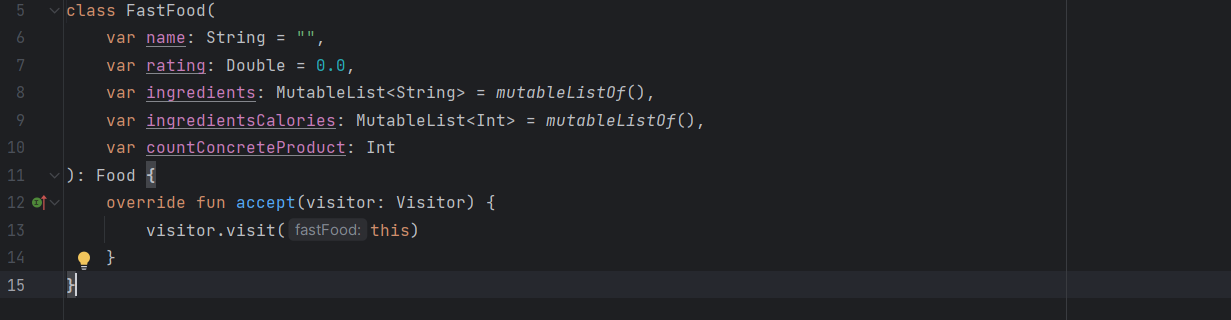
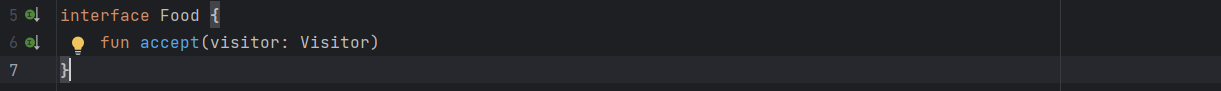


При разработке приложения были реализованы классы Salad и FastFood, представляющие собой различные типы блюд. Оба класса имплементируют интерфейс Food, который содержит методы для получения рейтинга, ингредиентов и веса блюда.

Однако, при анализе кода можно заметить дублирование кода в методах getRating и getIngredients у классов Salad и FastFood. Это приводит к проблеме поддержки кода, так как при изменении логики вывода рейтинга или ингредиентов придется вносить изменения в двух местах.

Для решения данной проблемы и улучшения поддерживаемости кода можно применить паттерн "Посетитель" (Visitor). Паттерн "Посетитель" позволяет вынести операции над объектами из иерархии классов в отдельные классы-посетители, что позволяет добавлять новые операции без изменения существующих классов.

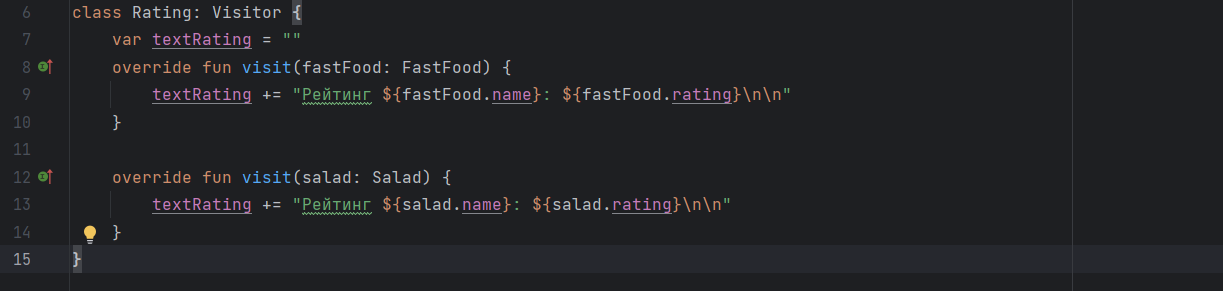
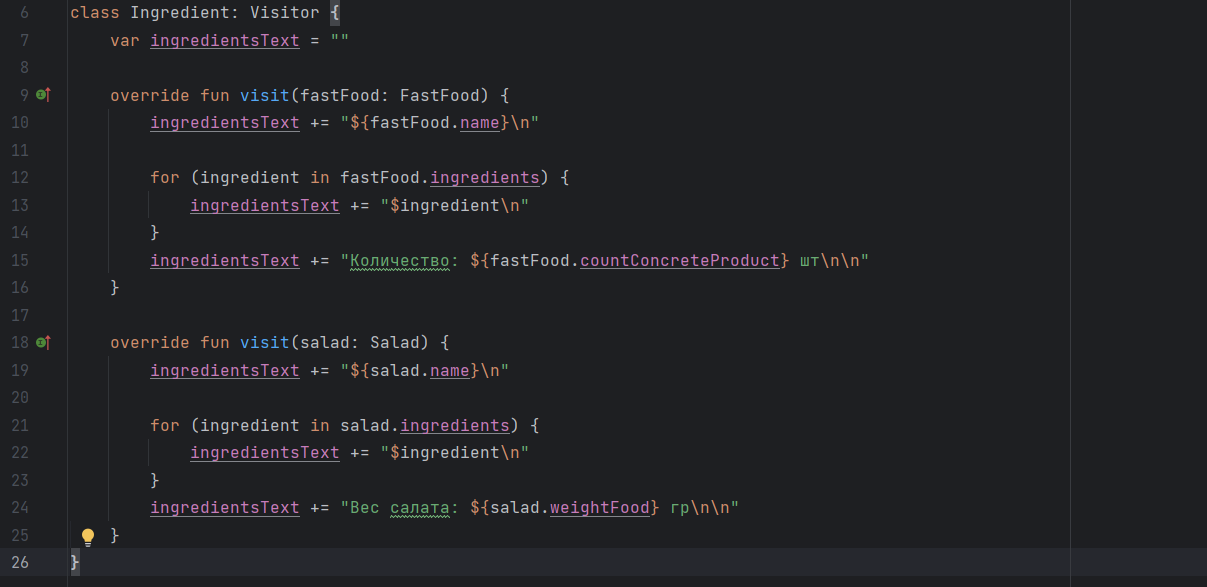
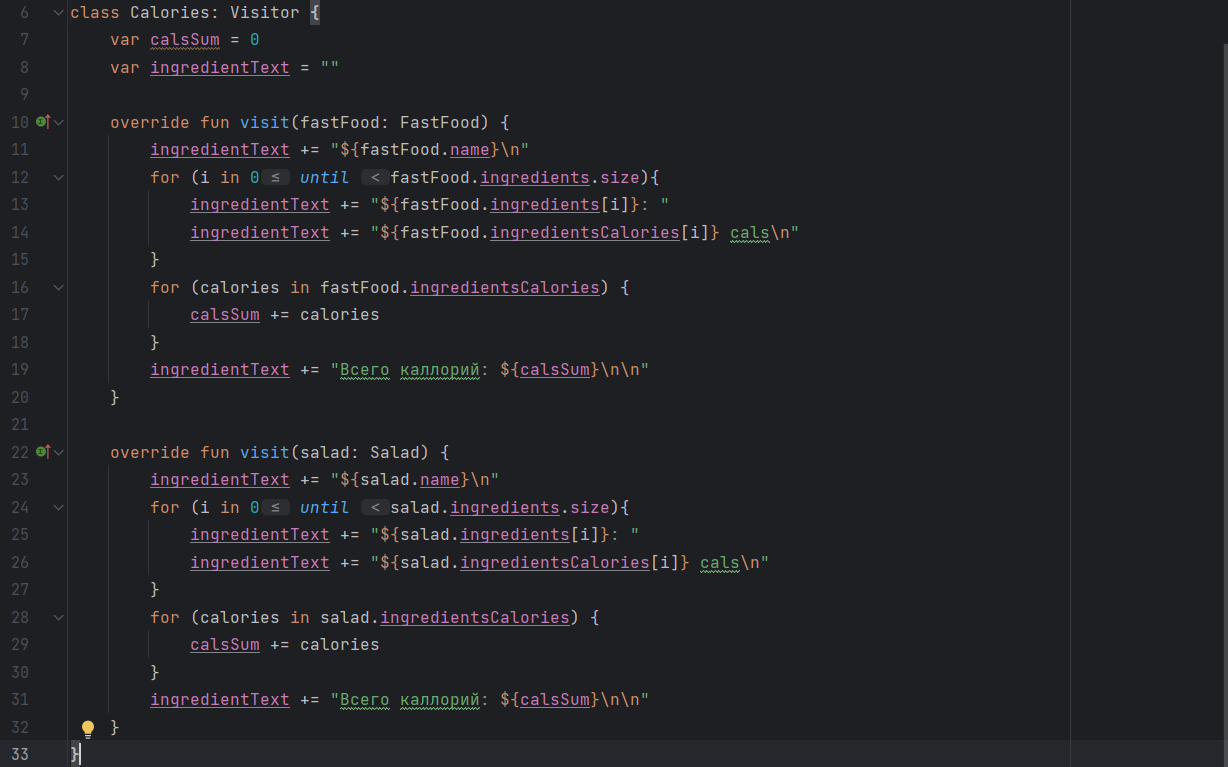
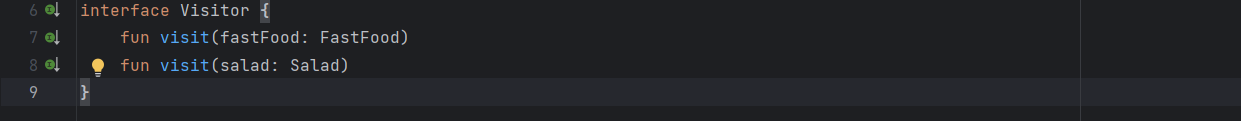
**Версия с паттерном**

****

1. Интерфейс Food и классы FastFood и Salad:

- Интерфейс Food определяет метод accept(visitor: Visitor), который позволяет объектам классов FastFood и Salad принимать посетителя.

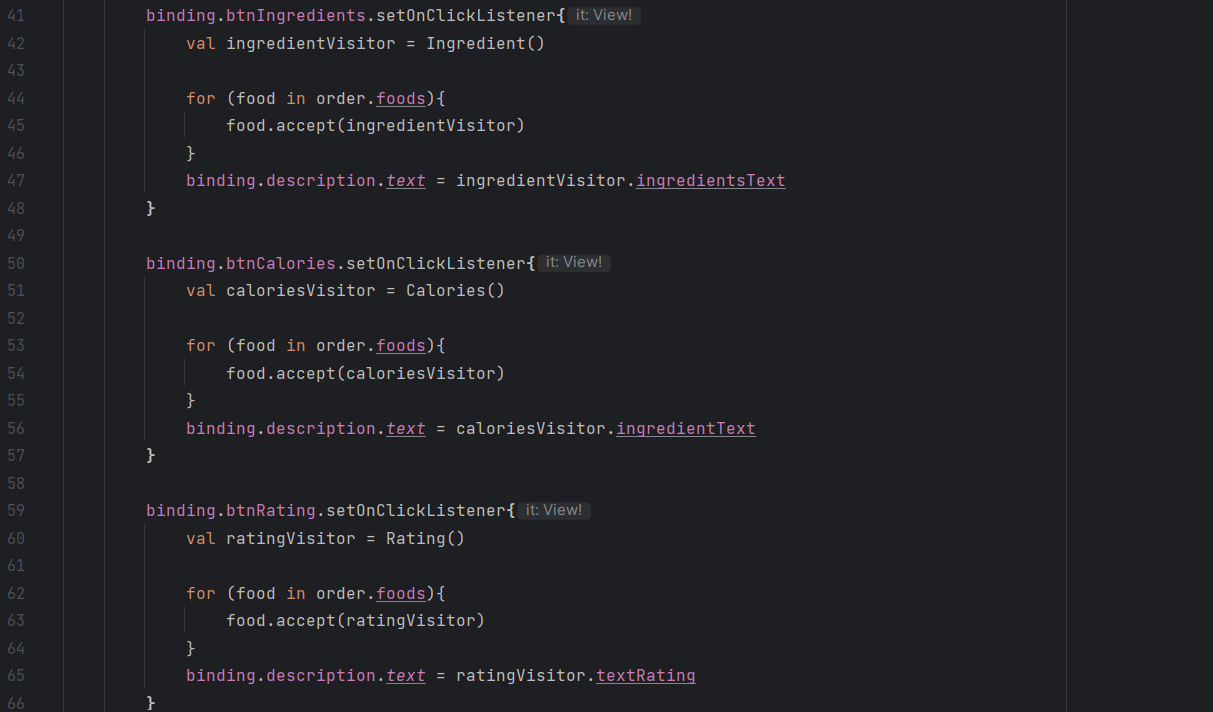
- Классы FastFood и Salad имплементируют интерфейс Food и реализуют метод accept(visitor), вызывая метод visit(this) посетителя.



2. Интерфейс Visitor и его реализации:

- Интерфейс Visitor определяет методы visit(fastFood: FastFood) и visit(salad: Salad), которые будут вызываться посещаемыми объектами.

- Реализации интерфейса Visitor, такие как Calories, Ingredient и Rating, представляют собой различные операции, которые можно выполнить над объектами классов FastFood и Salad.

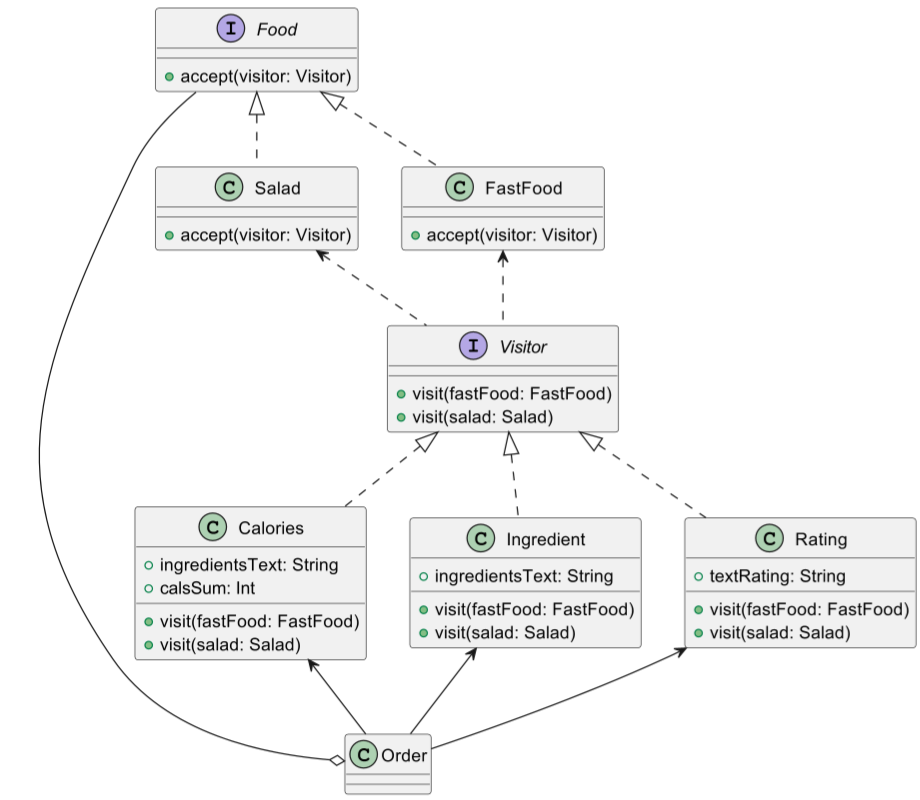


Применение паттерна в классе ItemFragment:

- В классе ItemFragment при нажатии на кнопки "btnIngredients", "btnCalories" и "btnRating" создаются соответствующие посетители (Ingredient, Calories и Rating) и применяются к объектам заказа.

- Посетители обходят все элементы заказа и выполняют операции над каждым из них в соответствии с их типом

**Диаграмма классов**

****

**Преимущества использования паттерна "Посетитель"**

- Избежание дублирования кода: Паттерн "Посетитель" позволяет вынести операции над объектами в отдельные классы-посетители, что избавляет от дублирования кода.

- Гибкость добавления новых операций: Добавление новых операций над объектами становится простым, так как не требуется изменять существующие классы.

**Заключение**

Паттерн "Посетитель" помогает улучшить структуру кода, сделать его более гибким и поддерживаемым. В данном случае, применение этого паттерна в классах FastFood и Salad позволяет выполнять различные операции над объектами еды без изменения самих классов, что способствует лучшей организации кода и его расширяемости.