Рефакторинг

Вадим Калашников
Lead Software Engineer
в компании Wildberries

Проверка связи

Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включён звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти

Поставьте в чат:

\+ если меня видно и слышно

— если нет

Вадим Калашников

О спикере:

- Разработчик на C++ более 15 лет
- Опыт разработки в областях: backend, embedded, kernel development, системное программирование, сети.
- С 2023 года Lead Software Engineer в компании Wildberries

Вспоминаем прошлое занятие

Вопрос: какая стратегия запускает задачу всегда асинхронно?

Вспоминаем прошлое занятие

Вопрос: какая стратегия запускает задачу всегда асинхронно?

Ответ: std::launch::async

Вспоминаем прошлое занятие

Вопрос: результат какого типа возвращается из асинхронной задачи?

Вспоминаем прошлое занятие

Вопрос: результат какого типа возвращается из асинхронной задачи?

Ответ: std::future

Вспоминаем прошлое занятие

Вопрос: как можно уведомить основной поток, что асинхронная задача выполнила свою работу?

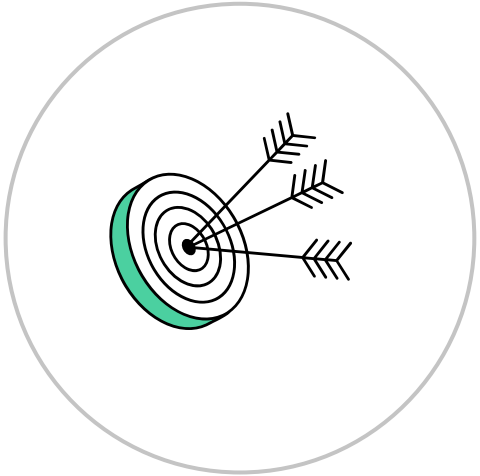
Вспоминаем прошлое занятие

Вопрос: как можно уведомить основной поток, что асинхронная задача выполнила свою работу?

Ответ: через использование шаблонов std::packaged_task<> и std::promise<T>

Цели занятия

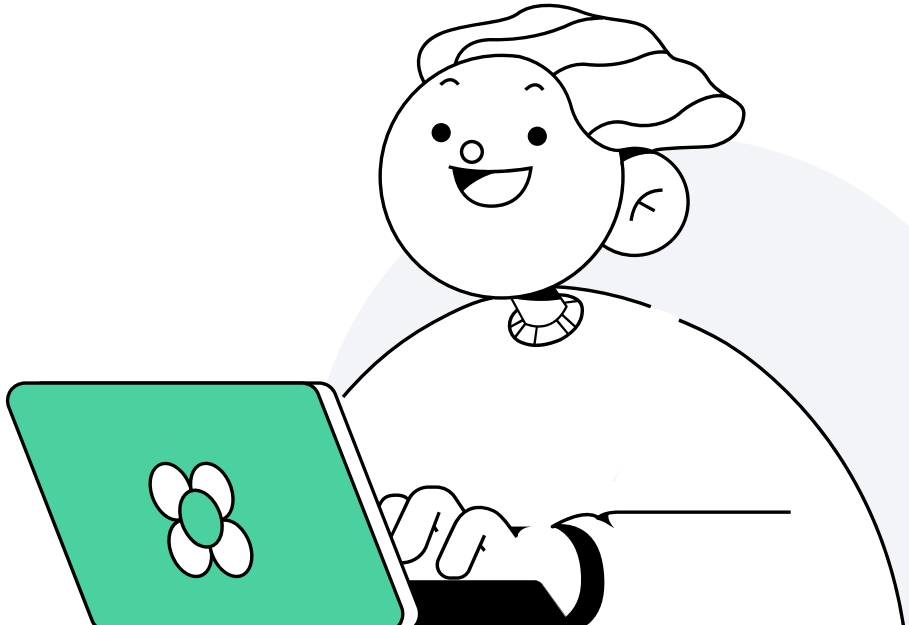
- Узнаем, что такое рефакторинг
- Разберём «плохие запахи» в коде
- Создадим код, свободный от «запахов»



План занятия

1. Основы рефакторинга
2. «Плохие запахи» кода
3. Пример рефакторинга
4. Итоги
5. Домашнее задание

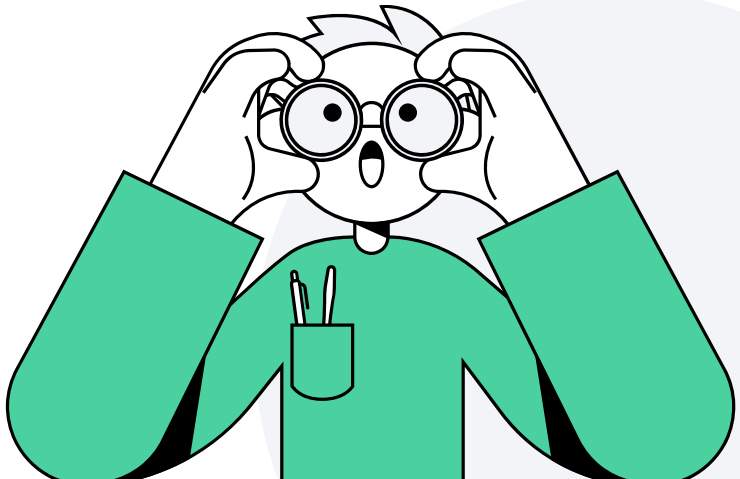
*Нажми на нужный раздел для перехода



Основы рефакторинга

1

Рефакторинг — упрощение понимания программного кода и его модификации в будущем



Оптимизация — достижение максимальной производительности, часто ценой удобного кода

Расширение функциональности — добавление новых функциональных возможностей, нового поведения

Цели рефакторинга

- Улучшает композицию программного кода
- Облегчает понимание программного кода
- Помогает найти ошибки
- Помогает писать код быстрее

«Плохие запахи» кода

2

«Плохие запахи» в коде

Дублирование кода

В разных методах существует код, выполняющий одни и те же действия

«Плохие запахи» в коде

Дублирование кода

В разных методах существует код, выполняющий одни и те же действия

Длинные методы

Один метод одновременно решает более одной задачи

«Плохие запахи» в коде

Дублирование кода

В разных методах существует код, выполняющий одни и те же действия

Длинные методы

Один метод одновременно решает более одной задачи

Большой класс

Класс пытается выполнять слишком много работы и вмещает в себя слишком много атрибутов

«Плохие запахи» в коде

Дублирование кода

В разных методах существует код, выполняющий одни и те же действия

Длинные методы

Один метод одновременно решает более одной задачи

Большой класс

Класс пытается выполнять слишком много работы и вмещает в себя слишком много атрибутов

Длинный список параметров

Метод или методы должны принимать слишком много параметров

«Плохие запахи» в коде

Дублирование кода

В разных методах существует код, выполняющий одни и те же действия

Длинные методы

Один метод одновременно решает более одной задачи

Большой класс

Класс пытается выполнять слишком много работы и вмещает в себя слишком много атрибутов

Длинный список параметров

Метод или методы должны принимать слишком много параметров

Расходящиеся модификации

Класс часто модифицируется различными способами по разным причинам

«Плохие запахи» в коде

Дублирование кода

В разных методах существует код, выполняющий одни и те же действия

Длинные методы

Один метод одновременно решает более одной задачи

Большой класс

Класс пытается выполнять слишком много работы и вмещает в себя слишком много атрибутов

Длинный список параметров

Метод или методы должны принимать слишком много параметров

Расходящиеся модификации

Класс часто модифицируется различными способами по разным причинам

Стрельба дробью

При выполнении любых модификаций приходится вносить множество мелких изменений в большое количество классов

«Плохие запахи» в коде

Завистливые функции

Метод некоторого класса чаще работает с данными внешних классов, чем со своими собственными

«Плохие запахи» в коде

Завистливые функции

Метод некоторого класса чаще работает с данными внешних классов, чем со своими собственными

Группы данных

Одна и та же связка данных встречается совместно хотя бы в нескольких сигнатурах методов

«Плохие запахи» в коде

Завистливые функции

Метод некоторого класса чаще работает с данными внешних классов, чем со своими собственными

Группы данных

Одна и та же связка данных встречается совместно хотя бы в нескольких сигнатурах методов

Временное поле

Атрибут некоторого объекта устанавливается только при определённых обстоятельствах

«Плохие запахи» в коде

Завистливые функции

Метод некоторого класса чаще работает с данными внешних классов, чем со своими собственными

Группы данных

Одна и та же связка данных встречается совместно хотя бы в нескольких сигнатурах методов

Временное поле

Атрибут некоторого объекта устанавливается только при определённых обстоятельствах

Оператор switch

Один и тот же блок switch встречается в программе несколько раз

«Плохие запахи» в коде

Завистливые функции

Метод некоторого класса чаще работает с данными внешних классов, чем со своими собственными

Группы данных

Одна и та же связка данных встречается совместно хотя бы в нескольких сигнатурах методов

Временное поле

Атрибут некоторого объекта устанавливается только при определённых обстоятельствах

Оператор switch

Один и тот же блок switch встречается в программе несколько раз

Ленивый класс

Существование некоторого класса не окупается выполняемыми им функциями

«Плохие запахи» в коде

Завистливые функции

Метод некоторого класса чаще работает с данными внешних классов, чем со своими собственными

Группы данных

Одна и та же связка данных встречается совместно хотя бы в нескольких сигнатурах методов

Временное поле

Атрибут некоторого объекта устанавливается только при определённых обстоятельствах

Оператор switch

Один и тот же блок switch встречается в программе несколько раз

Ленивый класс

Существование некоторого класса не окупается выполняемыми им функциями

Неуместная близость

Один класс знает о другом слишком много

«Плохие запахи» в коде

Неполнота библиотечного класса

Библиотечный класс сложно обучить новому поведению

«Плохие запахи» в коде

Неполнота библиотечного класса

Библиотечный класс сложно обучить новому поведению

Классы данных

Некоторый класс содержит лишь поля и методы доступа к ним

«Плохие запахи» в коде

Неполнота библиотечного класса

Библиотечный класс сложно обучить новому поведению

Классы данных

Некоторый класс содержит лишь поля и методы доступа к ним

Цепочки сообщений

Объект запрашивает данные у объекта, которых у него нет, и тот вынужден запрашивать их у другого объекта и т. д. вниз по иерархии

«Плохие запахи» в коде

Неполнота библиотечного класса

Библиотечный класс сложно обучить новому поведению

Классы данных

Некоторый класс содержит лишь поля и методы доступа к ним

Цепочки сообщений

Объект запрашивает данные у объекта, которых у него нет, и тот вынужден запрашивать их у другого объекта и т. д. вниз по иерархии

Посредник

Обнаруживается, когда интерфейс класса больше занят делегированием обязанностей, чем их реальным выполнением

«Плохие запахи» в коде

Неполнота библиотечного класса

Библиотечный класс сложно обучить новому поведению

Классы данных

Некоторый класс содержит лишь поля и методы доступа к ним

Цепочки сообщений

Объект запрашивает данные у объекта, которых у него нет, и тот вынужден запрашивать их у другого объекта и т. д. вниз по иерархии

Посредник

Обнаруживается, когда интерфейс класса больше занят делегированием обязанностей, чем их реальным выполнением

Отказ от наследства

Классы унаследуют от родительских классов больше, чем им требуется

«Плохие запахи» в коде

Неполнота библиотечного класса

Библиотечный класс сложно обучить новому поведению

Классы данных

Некоторый класс содержит лишь поля и методы доступа к ним

Цепочки сообщений

Объект запрашивает данные у объекта, которых у него нет, и тот вынужден запрашивать их у другого объекта и т. д. вниз по иерархии

Посредник

Обнаруживается, когда интерфейс класса больше занят делегированием обязанностей, чем их реальным выполнением

Отказ от наследства

Классы унаследуют от родительских классов больше, чем им требуется

Комментарии

Комментарии используют, чтобы скрыть «плохие запахи»

«Плохие запахи» в коде

Одержимость элементарными типами

Элементарные типы применяются там, где должны быть объекты

«Плохие запахи» в коде

Одержимость элементарными типами

Элементарные типы применяются там, где должны быть объекты

Параллельные иерархии наследования

При порождении экземпляра одного подкласса необходимо порождать экземпляр другого подкласса

«Плохие запахи» в коде

Одержимость элементарными типами

Элементарные типы применяются там, где должны быть объекты

Параллельные иерархии наследования

При порождении экземпляра одного подкласса необходимо порождать экземпляр другого подкласса

Альтернативные классы с разными интерфейсами

Методы выполняют одни и те же действия, но имеют различные сигнатуры

«Плохие запахи» в коде

Одержимость элементарными типами

Элементарные типы применяются там, где должны быть объекты

Параллельные иерархии наследования

При порождении экземпляра одного подкласса необходимо порождать экземпляр другого подкласса

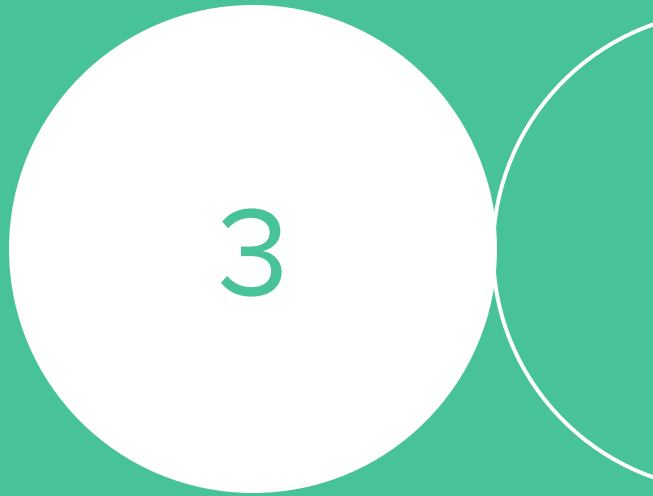
Альтернативные классы с разными интерфейсами

Методы выполняют одни и те же действия, но имеют различные сигнатуры

Теоретическая общность

Программы пытаются реализовать набор механизмов для работы с вещами, которые не нужны

Пример рефакторинга



Пример плохого кода

```
class Movie
{
	public:
		const int REGULAR = 0;
		const int NEW_RELEASE = 1;
		const int CHLDREN = 2;
		Movie(std::string _title, int _priceCode) {
			title = _title;
			priceCode = _priceCode;
		}
		void operator=(const Movie& value){
			title = value.title;
			priceCode = value.priceCode;
		}
		void setPriceCode(int value) { priceCode = value; }
		int getPriceCode() { return priceCode; }
		std::string getTitle() { return title; }
	private:
		std::string title;
		int priceCode = 0;
};
```

Пример плохого кода

```
class Rental
{
     public:
     Rental(Movie _movie, int _daysRented)
     {
          movie = _movie;
          daysRented = _daysRented;
     }
     Movie getMovie() { return movie; }
     int getDaysRented() { return daysRented; }

private:
     Movie movie;
     int daysRented = 0;
};
```

Пример плохого кода

```
class Customer
{
private:
    std::string name;
    std::vector<Rental> rentals;
public:
    Customer(std::string _name)
    {
        name = _name;
    }
    void addRentals(Rental rent)
    {
        rentals.push_back(rent);
    }
    std::string getName()
    {
        return name;
    }
```

Пример плохого кода

```
std::string Statement()
    {
        // Сначала мы объявляем локальные переменные.
        double totalAmount = 0;        int frequentRenterPoints = 0;
        std::string result = "Учет аренды для : " + name;
        // Затем для каждого клиента мы рассчитываем задолженность...
        for (auto rent : rentals) {
            double thisAmount = 0;
            // Определить сумму для каждой строки.
            switch (rent.getMovie().getPriceCode()) {
            case 0:
                thisAmount += 2;
                if (rent.getDaysRented() > 2)
                    thisAmount += (rent.getDaysRented() - 2) * 1.5;
                break;
            case 1:
                thisAmount += rent.getDaysRented() * 3;
                break;
```

Пример плохого кода

```
        case 2:
                thisAmount += 1.5;
                if (rent.getDaysRented() > 3)
                    thisAmount += (rent.getDaysRented() - 3) * 1.5;
                break;
            }
            // Добавить очки для активного арендатора.
            frequentRenterPoints++;
            // Бонус за аренду новинки на два дня.
            if ((rent.getMovie().getPriceCode() == 1) && rent.getDaysRented() > 1)
                frequentRenterPoints++;
            // Показать результаты для этой аренды
            result += "\t" + rent.getMovie().getTitle() + "\t" + std::to_string(thisAmount) + "\n";
            totalAmount += thisAmount;
         }
         // Добавить нижний колонтитул
         result += "Сумма задолженности составляет " + std::to_string(totalAmount) + "\n";
         result += "Вы заработали " + std::to_string(frequentRenterPoints) + " очков ";
         return result;    }
};
```

Итоги

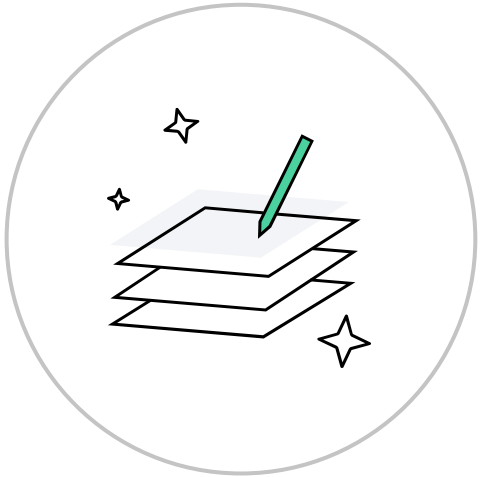
Итоги занятия

1. Узнали, какие бывают «плохие запахи»
2. Разобрали, как бороться с «плохими запахами»
3. Очистили код от «запахов»

Домашнее задание

Давайте посмотрим ваше домашнее задание

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Дополнительные материалы

- Мартин Фаулер «Рефакторинг: улучшение существующего кода»

Задавайте вопросы и пишите отзыв о лекции