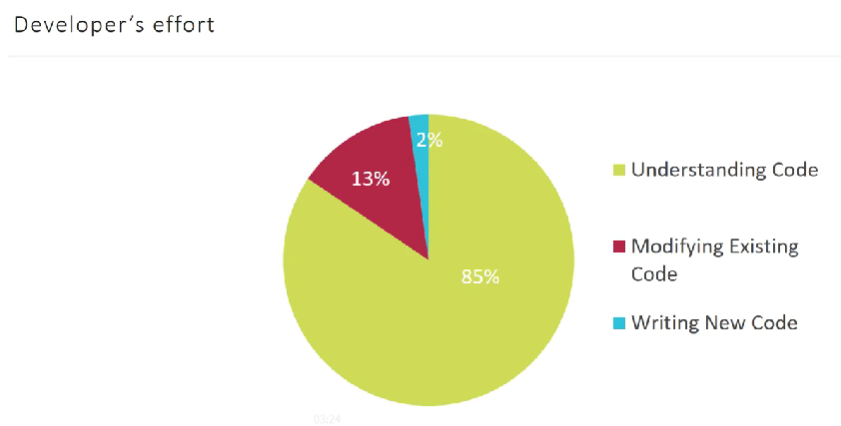
**Сlean code concept**

**Определение чистого кода. Почему чистый код так важен?**

****

**Преимущества, если будем писать чистый код:**

* Быстрое чтение;
* Можно быстро исправить;
* Менее дорогой;
* Менее сложный;
* Меньше багов;
* Выглядит профессиональным.

**Поддержание чистоты кода:**

* Сделать это целью команды;
* Регулярная задача (регулярно работать над чистотой кода);
* Использование каких-то стандартов;
* Определенная практика;
* Инструменты.

**Рефакторинг** — **это** процесс изменения кода, призванный упростить его обслуживание, понимание и расширение, при этом не изменяя его поведение.

**Code convention – соглашение о коде**

**Помощь в улучшении чистоты кода.**

Рекомендация о то, как лучше всего на этом языке описать какие-то вещи с точки зрения синтаксиса.

**Naming variables**

**Рекомендации:**

* использовать Code Convention;
* понятные имена;
* если однобуквенное значение, то I и j;
* если константа, то ее название должно быть написано капсом (ALL\_CAPS);

**Че стоит избегать:**

* переменные с абстрактными и очевидными названиями (status, result)
* венгерскую нотацию – есть переменная с префикосм (mArray); Венгерская нотация является одной из методик, позволяющих программистам создавать более читаемый код в короткие сроки.
* магических чисел;
* аббревиатуры;

**Naming classes**

**Рекомендации:**

* название должно быть существительным;
* исчерпывающие имена;

**Че стоит избегать:**

* God class/Magic Hammer. Класс использует антипаттерн. Когда класс содержит много назначений (папка с мусором);
* Аббревиатуры;
* Неочевидных названий;
* Использование цифр;
* Название «MyClass».

**Naming functions**

**Рекомендации:**

* название должно говорить о том, что он делает; Название не может быть существительным;
* длинные имена.

**Че стоит избегать:**

* не делать 2 разных действия в 1 методе. 1 метод – 1 задача;
* непонятные “check….”, “try….”;
* void get(), string print() etc.;
* больше 3ех параметров.

**Comments**

**Рекомендации:**

* указывание лицензии;
* сложные комментарии в стиле джава-документа, пишем библиотеку;
* писать в коммах причину

**Че стоит избегать:**

* не использовать /\* …. \*/;
* не коммитить в гит комментарии //todo (это служебные комментарии);
* не писать очевидные комментарии;
* комментарии на закрытие скобок цикла;
* лучший способ сохранить код – закоммитать на гит, нежели в коде;
* не каждый метод должен иметь комментарии

**Excess code – избыточный код**

**Рекомендации:**

* если хотим что-то вернуть - сразу возвращаем
* используем тернарный оператор.

**Че стоит избегать:**

* System.out.println(). Консоль никто не читает.
* Не сравнивать булиан с булианом. Использовать AreTrue/False
* Не надо создавать переменную, только чтобы ее вернуть

**Exceptions (исключения)**

**Рекомендации:**

* стоит использовать;
* название исключения должно соответствовать той ошибке, которую мы получили.

**Че стоит избегать:**

* не возвращать какие-то константы;

**Tests**

Тест должен быть линейным.

Не запускать тесты в main

Кроме методов ничего не должно быть

Не передавать параметры в тесты

Не нумеровать тесты по порядку

Не делать if else

**Рекомендации:**

* Ассерт должен быть
* Один тест – одна проверка
* Правильные и хорошие названия
* Тест должен запускаться в любом порядке, независимо от других

**Че стоит избегать:**

* Не указывать в названии тестовые данные
* Не делать if else/ for while и тд
* Не выстраивать цепочку запускания тестов

**Design principles**

**Рекомендации:**

* KISS **(Keep it Simple and Straightforward)** – делать просто;
* DRY **(Don’t Repeat Yourself)** – не повторяйтесь;

**Че стоит избегать:**

* Слепое копирование
* Написать в main, потом разбить на классы
* Не оставлять на потом, сразу делать чистый код

**Check list & summary**

* Инженер читает больше кода, чем пишет
* Наименование переменных, классов, методов очень важно. Это ускоряет чтение кода и экономит время.
* Использование комментариев только там, где это необходимо.
* Стараться снижать сложность
* KISS and DRY
* Делать собственные исключения
* Использовать инструменты для чистого кода. Экономия времени
* Быть инженером, а не кодером
* Стоить читать книги

