Что такое качественный код, и как его писать?

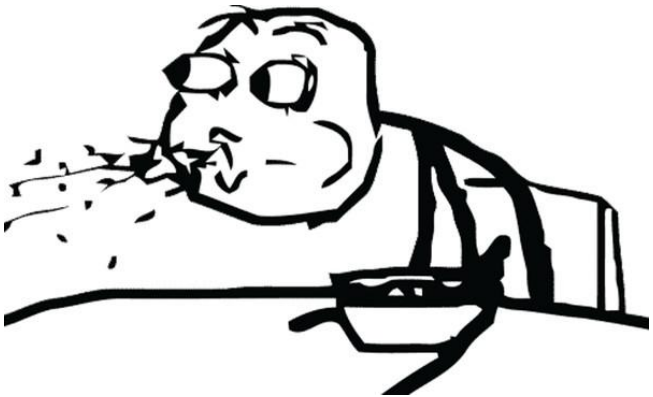
Федор Юданов

Senior Software Engineer @ Xored,
Выпускник ФИТ НГУ

Что нужно знать профессиональному программисту?

- Языки программирования (Java, C++, JS...)
- Алгоритмы *($O(n\log n)$, P=NP?, вычислимость...)*
- Структуры данных *(списки, деревья, хэш-таблицы...)*
- Принципы ООП (полиморфизм, абстракция, наследование...)
- Принципы дизайна (ORR, LSP, Rule of Demeter...)
- Паттерны проектирования (singleton, observer, factory...)

Что нужно знать профессиональному программисту?



Обо всем этом мы сегодня говорить не будем!

Сегодня мы обсудим

1. Что такое качественный код?
2. Что такое красивый код?
3. Как писать красивый и качественный код?
4. Как профессионалы работают над качеством кода?
5. А так ли нам нужен качественный код?

Каждый дурак может написать код, понятный компьютеру. Хорошие программисты пишут код, понятный людям.

— Мартин Фаулер

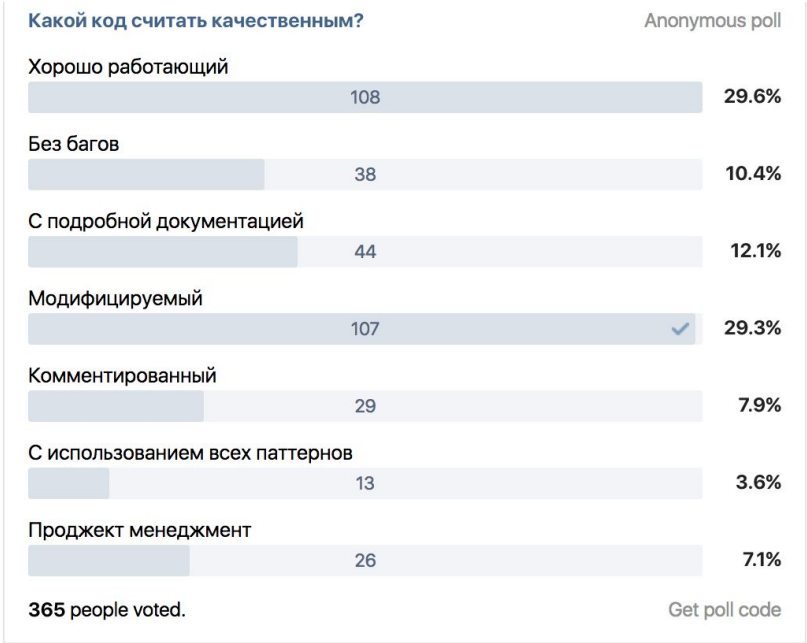
Пишите код, исходя из того, что все программисты, которые будут сопровождать вашу программу, — склонные к насилию психопаты, знающие, где вы живёте.

— Джон Ф. Вудс

Наиболее читаемый код — тот, который не написан.

— Дастин Босуэлл, Тревор Фаучер

Результаты опроса



Что такое качественный код?

- Хорошо работающий, без багов
- Легко читаемый
- Легко модифицируемый

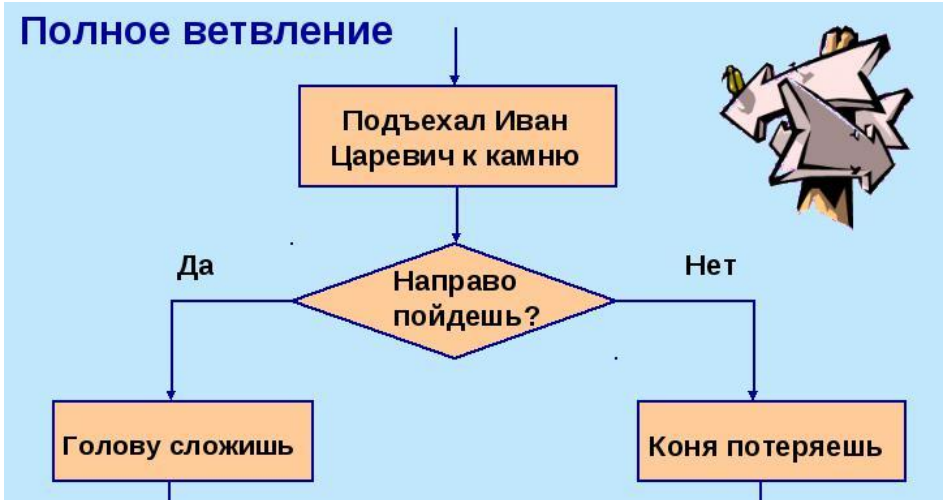
Что такое красивый код?

- Легко читаемый
- Легко модифицируемый
- *Допишите на свой вкус...*

3 главных качества легко читаемого кода

1. Линейный
2. Короткий
3. Самодокументированный

Линейность кода

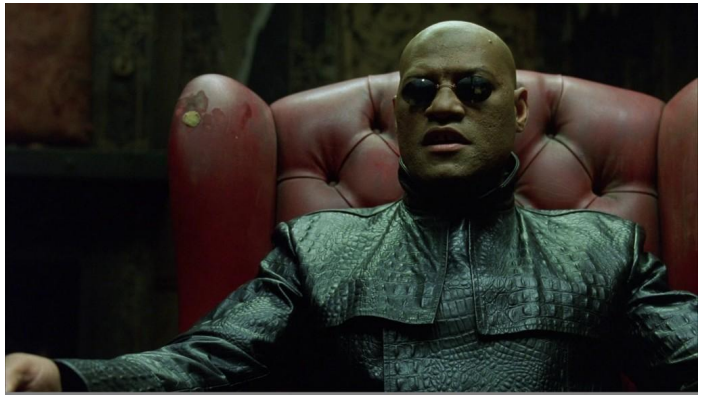


Линейность кода

```
public static String doStuff(String str) {
  if (str != null) {
    if (str.endsWith(";")) {
      int pos = str.length() - 1;
      while (pos >= 0) {
        if (str.charAt(pos) == ';')
          break;
        else
          pos--;
      }

      if (pos >= 0)
        return str.substring(0, pos);
      else
        return "";
    } else
      return str;
  }
  return null;
}
```

Welcome to the real world, Neo!



http://govnokod.ru/21229

Линейность кода

```
if (str != null) {
  if (str.endsWith(";")) {
    int pos = str.length() - 1;
    while (pos >= 0) {
      if (str.charAt(pos) == ';')
        break;
      else
        pos--;
    }

    if (pos >= 0)
      return str.substring(0, pos);
    else
      return "";
  } else
    return str;
}
return null;
```

```
if (str == null) return null;
if (!str.endsWith(";")) return str;

int pos = str.lastIndexOf(";");
return pos >= 0 ? str.substring(0, pos) : "";
```

Линеаризация кода

1. Выделяем главную ветвь алгоритма
2. Выносим ее на наименьший уровень вложенности
3. Повторяем с подветвями рекурсивно

Линеаризация кода

```
if (str != null) {
  if (str.endsWith(";")) {
    int pos = str.length() - 1;
    while (pos >= 0) {
      if (str.charAt(pos) == ';')
        break;
      else
        pos--;
    }

    if (pos >= 0)
      return str.substring(0, pos);
    else
      return "";
  } else
    return str;
}
return null;
```

```
if (str == null) return null;
if (!str.endsWith(";")) return str;

int pos = str.length() - 1;
while (pos >= 0) {
  if (str.charAt(pos) == ';') break;
  pos--;
}

if (pos >= 0)
  return str.substring(0, pos);
else
  return "";
```

Линеаризация кода

- Выносим логически целостный блок в отдельную процедуру
- Заменяем if тернарным оператором

```
if (str == null) return null;
if (!str.endsWith(";")) return str;

int pos = str.length() - 1;
while (pos >= 0) {
  if (str.charAt(pos) == ';') break;
  pos--;
}

if (pos >= 0)
  return str.substring(0, pos);
else
  return "";
```

```
if (str == null) return null;
if (!str.endsWith(";")) return str;

int pos = str.lastIndexOf(";");
return pos >= 0 ? str.substring(0, pos) : "";
```

или `return StringUtils.removeEnd(str, ";");`

Краткость кода

- Избегаем велосипедов
- Избегаем дублирования
- Удаляем неиспользуемый код

Нет кода, нет проблемы!

Бритва Оккама

```
int sum = countSum();
int increasedSum = sum + 1;
operate(increasedSum);
```

⇩

```
int sum = countSum();
operate(sum + 1);
```

⇩

```
operate(countSum() + 1);
```

Раньше философские дискуссии проходили бодрее...

Самодокументированность кода

- Обдуманно выбираем идентификаторы
- Избегаем появления синонимов и омонимов
- Предпочитаем явное неявному, прямое — косвенному

Самодокументированность кода

```
public void handleOutput(OutputStream os) {
    …
}
```

или

```
public void formatAndPrintOutput(OutputStream os) {
    …
}
```

```
public class Person {
    public static Person createSubject(String name) {
      …
    }
}
```

```
public class Person {
    public static Person createPerson(String name) {
      ...
    }
}
```

```
if (i == Math.abs(i)) {
   System.out.println(i);
}
```

```
if (i >= 0) {
   System.out.println(i);
}
```

Нужны ли комментарии?

```
/**
* Name
*/
private String name;
```

```
public String getStartupMessage() {
   // If this happens, somebody's been screwing around with the
database definitions andhas removed the restriction that a given
alarm may have only one entry in the
   return
   /*for (String str : inputStrings) {

       if (str != null) {
           result.append(str);
       }
   }*/


   // If we know for a fact that an alarm isn't getting  aggressively
about it until it gets fixed 1111!!!!
   "Hello world!"; //TODO: improve
}
```

```
/**
* returns true
* @return true
*/
private boolean isPasswordRequired() {
   return false;
}
```

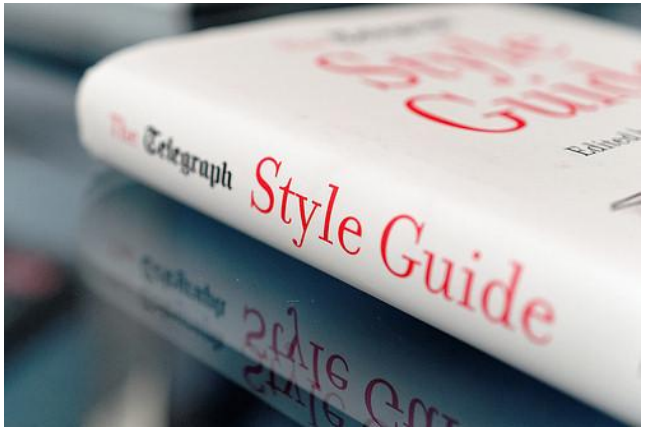
Когда комментарии полезны?

- Когда написать код понятно не получается
- Для спецификации контракта класса или метода
- Когда вы документируете библиотеку

Комментарии должны пояснять ЧТО и ПОЧЕМУ делает программа, а не КАК она это делает

Стили кодирования

- Проектные
- Стандартные



Google: java code conventions, венгерская нотация

Используйте автоформаттер.

Полагайтесь не “на глаз”, а на профессиональные инструменты!

Как еще повысить качество кода?

- Статические анализаторы
- Покрытие юнит-тестами
- Continuous Integration
- Code review
- Продуманная заранее архитектура

Следите за ~~новостями~~
нашими докладами!

“Альтернативная” точка зрения

- С чего вы взяли, что кто-то будет читать мой код?!!
- Да кому нужен этот код? Главное, что работает!!!
- Опытный программист быстро разберется в любом коде!!!

Что еще почитать?

1. https://habrahabr.ru/post/266969/ (Google: красивый код)
2. Steve McConnell. Code Complete.
3. Joshua Bloch. Effective Java.

Подведем итоги

1. Качественный и красивый код — почти одно и то же
2. Пишем код: линейный, короткий, самодокументированный
3. Профессионал владеет методами и инструментами, а не надеется на свою безошибочность
4. Вложил час в качество сейчас — сэкономил месяц в будущем

Спасибо за внимание!

Вопросы?