THE BEAUTY OF THE CODE

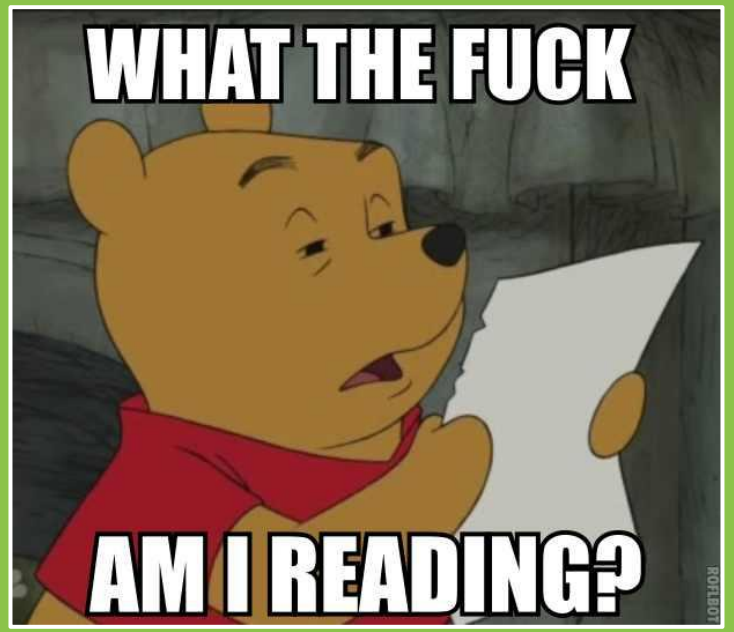
By Lidiya Georgieva

A brief note on clean code and code smells

INSPIRATION

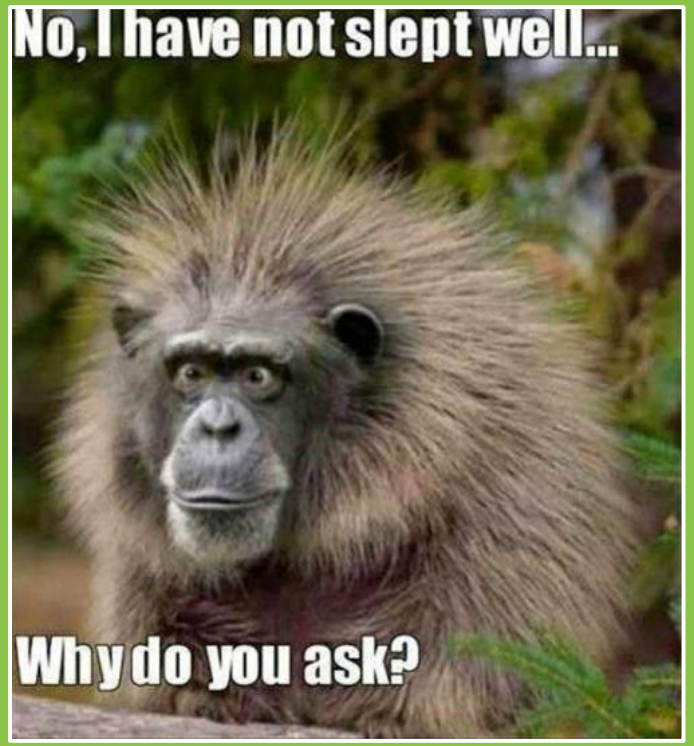
Why code quality matters?

Reading code for **hours** and still don't get it?

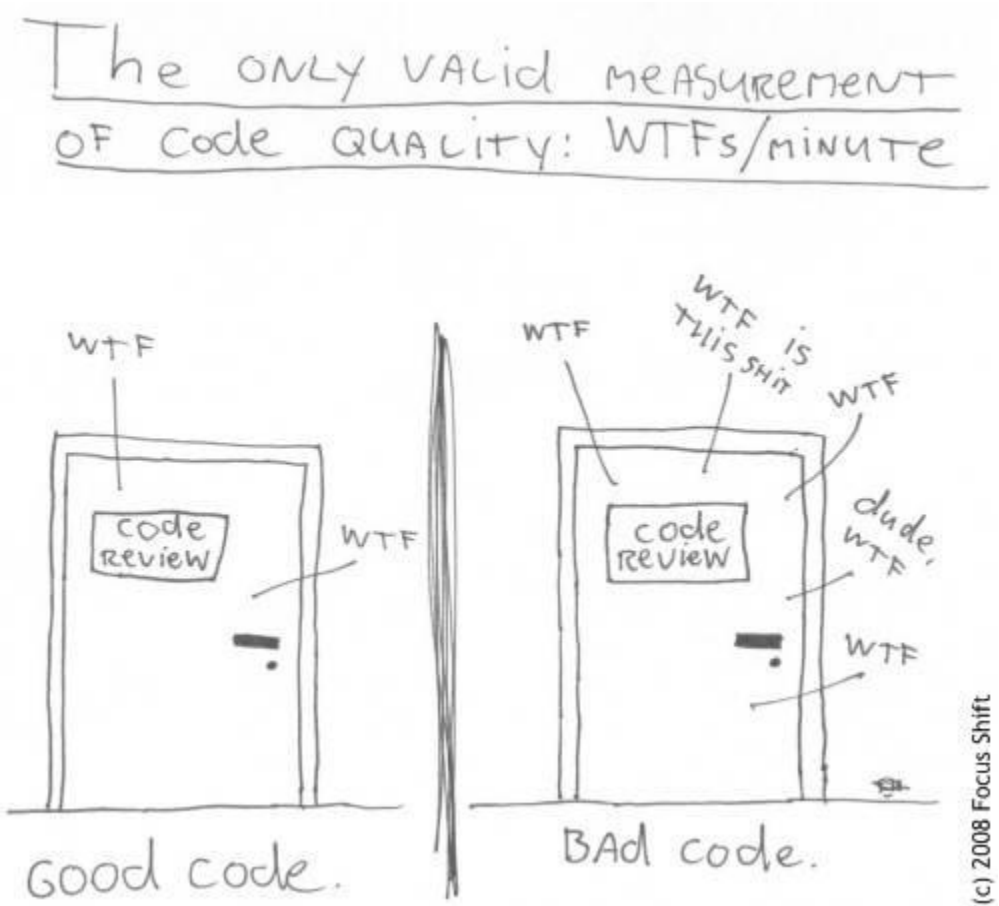


Changed some part of the system but another **breaks**?

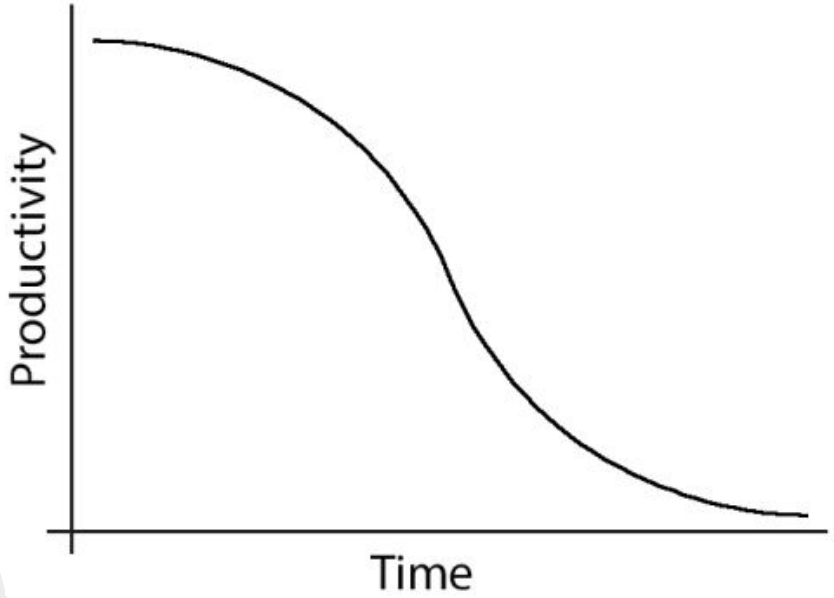
Don’t want to work on a certain ugly and tangled part of a project?



In which room are you?



Does your performance change when working on legacy code?



CODE SMELLS

Does your code stink?

Code smells

- Bad comments
- Bad functions
- Long lines
- Dead code
- Error handling

Code smells

- Artificial coupling
- Meaningful names
- Duplicated code

And many many more...

MEANINGFUL NAMES

Do your names require a comment?

Intention-revealing names

```
long schTime; // in milliseconds
```

vs

```
long scheduledTimeInMillis;

long bid = 123L;
```

vs

```
long brandId = 123L;
```

Something seems wrong

Intention-revealing names

```
String dsc = “”;
                    VS
String description = “”;

boolean bShouldCollect = true;
                    VS
boolean shouldCollect = true;
```

Ah.. that’s better

Sometimes funny, sometimes not

```
// because 42 is the answer to the ultimate
// question of Life, Universe and Everything
int d = 42;
```

Hint: Calendar

Verbs for methods

conditionsMet() { … }

vs

areConditionsMet() { … }

Nouns for classes

```
class GenerateToken { … }
```

vs

```
class TokenGenerator { … }
```

Meaningful in self context

```
class User {
      String nativeCity;
      String nativeZip;
      String currentCity;
      String currentZip;
}
```

Meaningful in self context

```
class Address {
      String city;
      String zip;
}
class User {
      Address nativeAddress;
      Address currentAddress;
}
```

FUNCTIONS

How long a function can be?

One thing

```
int res = UserPersister.insert(user);
if (res == 1) { .. }
else {...}
```

vs

```
try {
      UserPersister.insert(user);
} catch (SomeException ex) {..}
```

No side effects

```
public boolean createAccount(User user) {
    boolean success = AccountPersister.insert(user);
    MailSender.send(user.email,
        “Welcome on board, ” + user.name);
    updateStats();
    return success;
}
```

Many parameters

```
createAccount(String username, String
firstName, String lastName, String city,
String street, String state, String zip);
```

vs

```
createAccount(User user);
```

Flag parameters

```
getComments(true);


public List<Comment> getComments(boolean
getNonOwned) {
      if (getNonOwned) {
      // get users and owner coments
      } else { // get owner comments }
}
```

COMMENTS

Why certain comments are bad smell?

Failure to express in code

```
private long expiresIn; // in seconds

/** Return value is in seconds. */
public long getExpiresIn() {
    return expiresIn;
}

private long scheduleDelay; // seconds
```

Rewrite, do not add a comment

```
// whether we should get more comments
if (!comments.data.isEmpty &&
    comments.count != comments.data.length &&
    since < getLastTime(comments.data))
```

Compare this ...

Rewrite, do not add a comment

```
if (shouldGetMoreComments(comments, since)) {
      ....
}
```

.. and this. Do we need a comment now?

Commented out code

```
// private static final ConsoleLogger log =
//   new ConsoleLogger(OAuthHandler.class);
// private static final Logger log =
// Logger.getLogger(OAuthHander.class);
```

Long live version control systems

Commented out code

```
/**
 * This method is not used and will be
removed
 */
public List<Item> getAll() …


// TODO delete this method
```

Closing brace comments

```
        ........
} // class Data


        .......
} // if (addComments)
```

TL; DR

Noise Comments

```
/**
 * Returns the protocol.
 * @return The protocol.
 */
public Protocol getProtocol() {
      return this.protocol;
}
```

Noise Comments

```
/** The version.*/
private String version;

/** The version.*/
private String licenceName;

private String gender;// gender
```

REASONS

Why we happen to write bad code?

Why?

- Always in a rush
- Changing requirements
- Bad estimation
- Poor processes
- Lack of knowledge about clean code

Ways for Improvement

Where and how to start writing clean code?

How to improve

- Learn
- Practice
- Make code reviews
- Use helpful tools
  - sonarqube
- Write unit tests

RESOURCES

Eager to learn more?

Robert Martin (Uncle Bob) - *“Clean Code: A Handbook of Agile Software Craftsmanship”*

or https://cleancoders.com/videos

Martin Fowler - *”Refactoring: Improving the Design of Existing Code”*

Michael Feathers - *”Working Effectively with Legacy Code”*

FINAL WORDS

Any rules of thumb?

“

Leave the compound **better** than you found it.

The Boy Scout rule

Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live.

THANKS!

Questions?

You can find me

on Linkedin
https://bg.linkedin.com/in/lidiya-georgieva-156977a7