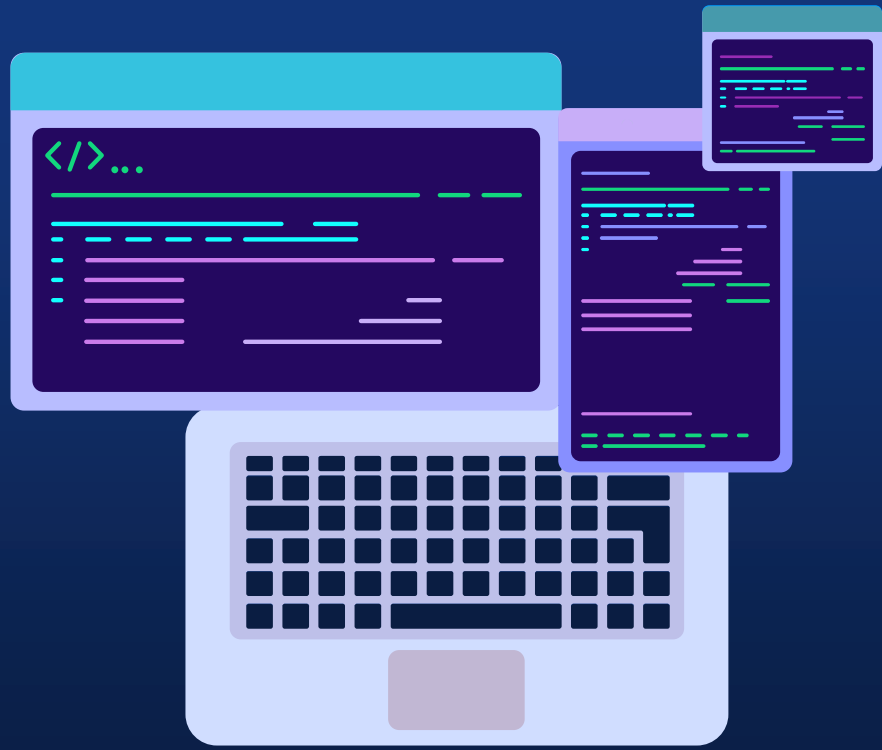
Software Design and Important concepts



Mentor: Einar Rocha

CONTENT

01
OOP Pillars

Inheritance, Polymorphism
Encapsulation, Abstraction

02
Clean Code

Meaningful Names,
Functions, Unit test
Code Smells...

03
SOLID

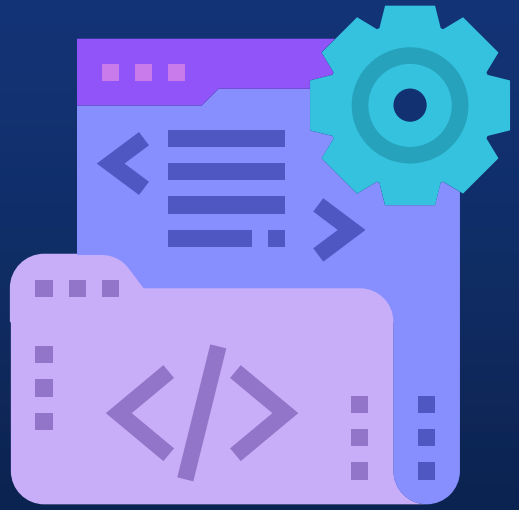
Single Responsiblity
Open closed
Liskov Substitution
Interface Segregation
Dependency Inversion

04
Design patterns

Singleton, Factory Method
Strategy, Observer
Builder...

02

Clean Code



The Goals of Software Design

- To allow us to write software that is as helpful as possible.
- To allow our software to continue to be as helpful as possible.
- To design systems that can be created and maintained as easily as possible by their programmers

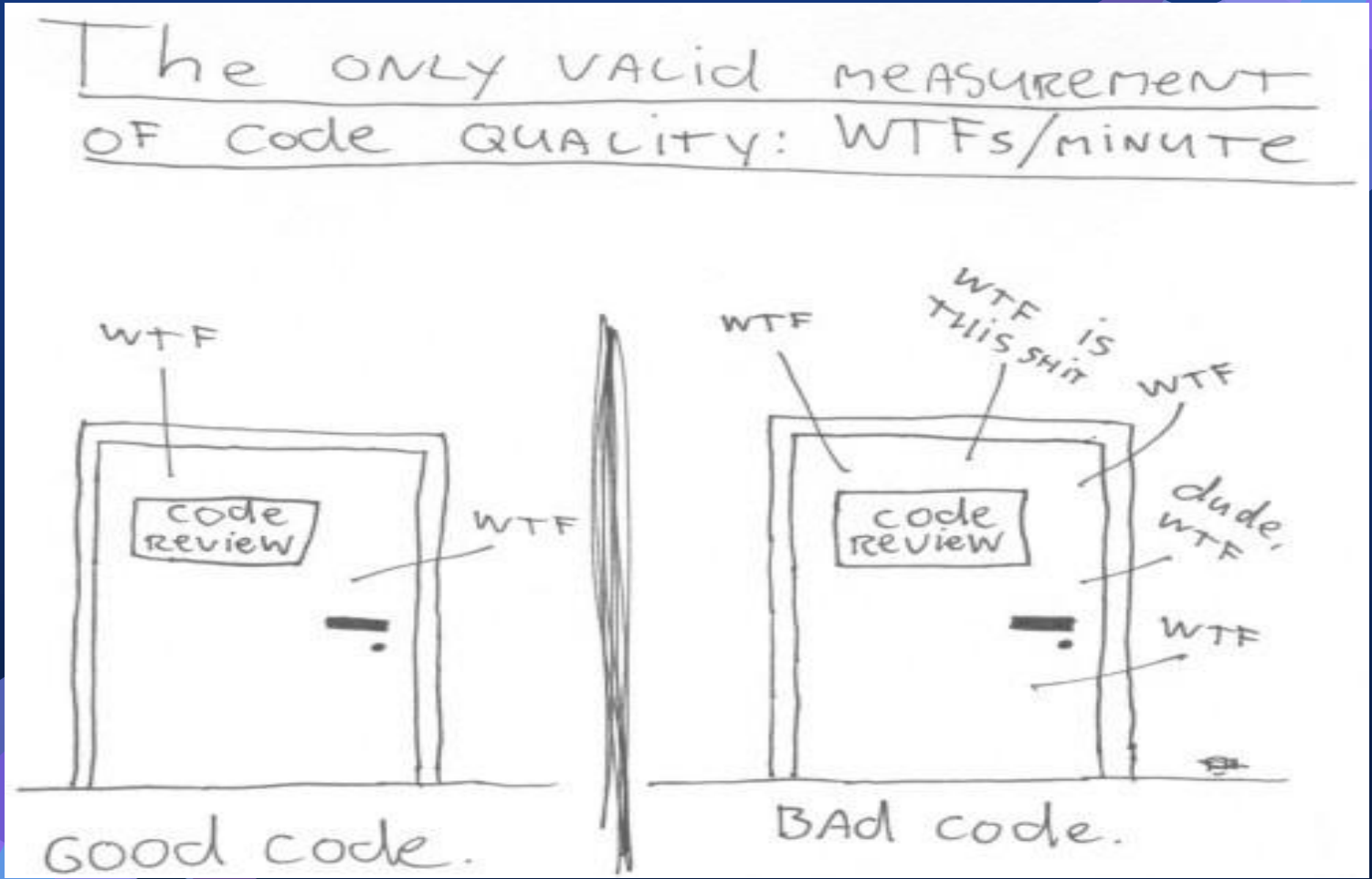
Agenda

Clean Code

Introduction
What Is Clean Code?
The Boy Scout Rule...

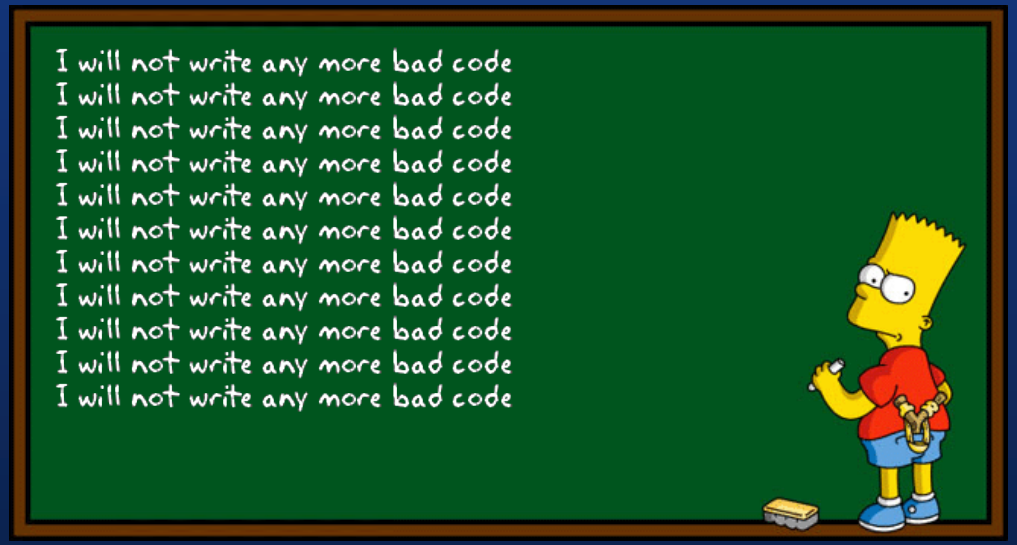
Meaningful Names

Disinformation, Encodings, Method Names, Meaningful Context...

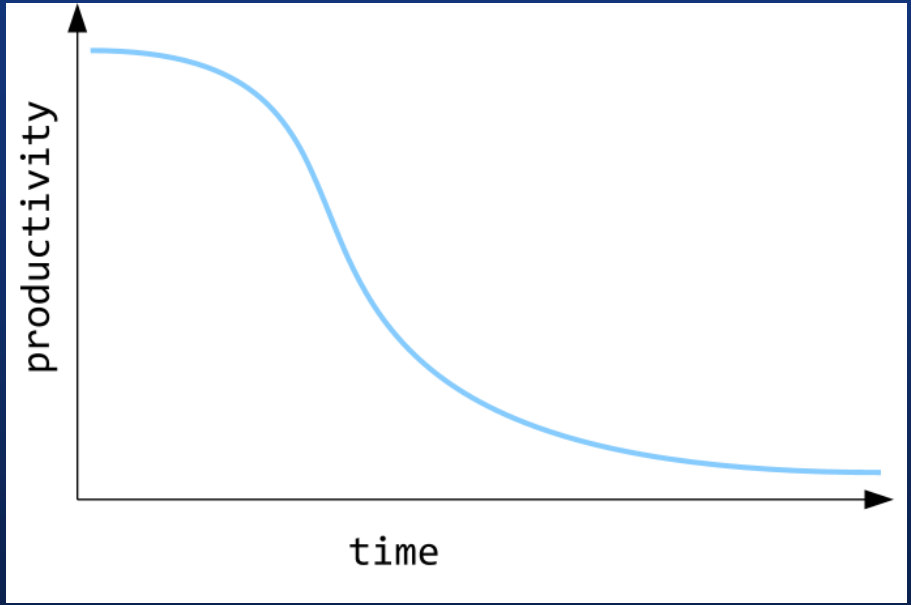

The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
Bad code.

There Will Be Code

Bad Code

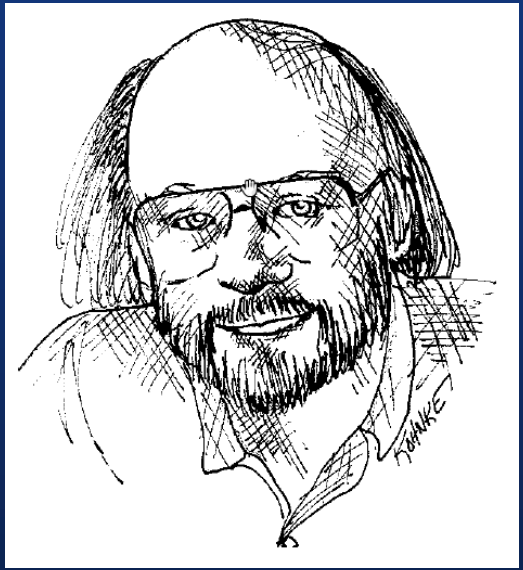

I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code

The Total Cost of Owning a Mess



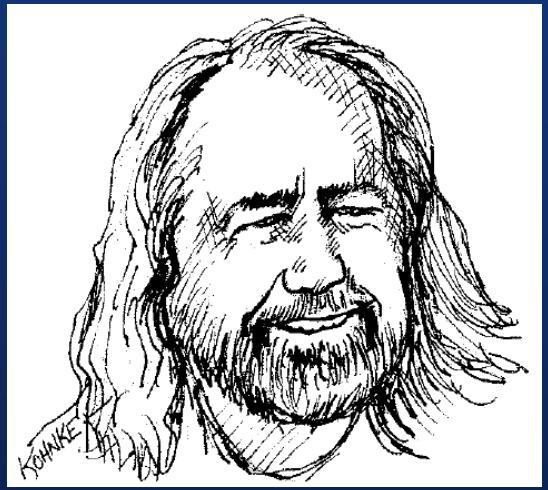
Clean Code?

There are probably as many definitions as there are programmers.



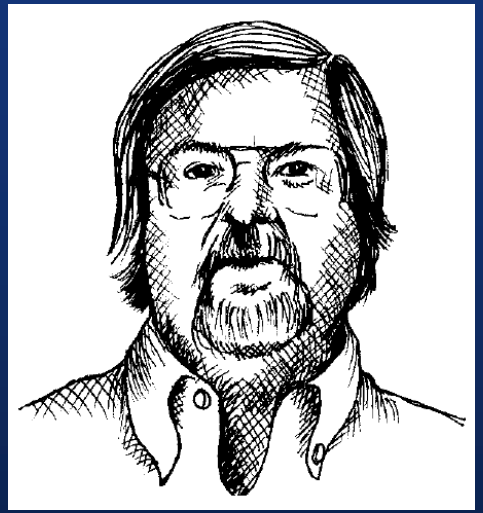
Bjarne Stroustrup, inventor of C++

…elegant and efficient.
…ease maintenance,
….performance close to optimal
….Clean code does one thing well.



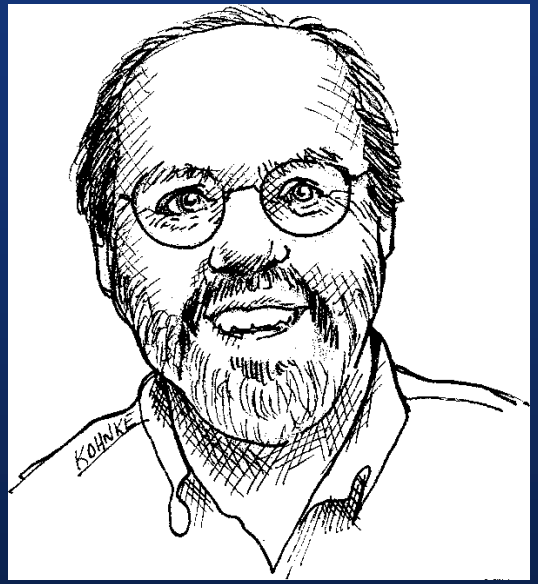
“Big” Dave Thomas, founder of OTI

Clean code can be read, and enhanced by a developer other than its original author. It has unit and acceptance tests. It has meaningful names. It provides one way rather than many ways for doing one thing. It has minimal dependencies,
which are explicitly defined, and provides a clear and minimal API……



Ron Jeffries, author of Extreme Programming

Runs all the tests;
Contains no duplication;
Expresses all the design ideas that are in the system;



Ward Cunningham, coinventor of eXtreme Programming.

You can call it beautiful code when
the code also makes it look like the language was made for the problem.

The Boy Scout Rule

Leave the campground cleaner than you found it

Meaningful Names

They are everywhere

Use Intention-Revealing Names

```
/**
 * Bad
 */
int d; // elapsed time in days

/**
 * God
 */
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

Use Intention-Revealing Names

```
public List<int[]> getThem() {
    List<int[]> list1 = new ArrayList<>();
    for (int[] x : theList)
        if (x[0] == 4)
            list1.add(x);
    return list1;
}
```

Use Intention-Revealing Names

```java
// Better
public List<int[]> getFlaggedCells() {
    List<int[]> flaggedCells = new ArrayList<int[]>();
    for (int[] cell : gameBoard)
        if (cell[STATUS_VALUE] == FLAGGED)
            flaggedCells.add(cell);
    return flaggedCells;
}

// Much Better
public List<Cell> getFlaggedCells() {
    List<Cell> flaggedCells = new ArrayList<Cell>();
    for (Cell cell : gameBoard)
        if (cell.isFlagged())
            flaggedCells.add(cell);
    return flaggedCells;
}
```

Avoid Disinformation

```
//poor variable names
int hp, aix, and, sco;



//names which vary in small ways
String XYZControllerForEfficientHandlingOfStrings;

String XYZControllerForEfficientStoragesOfStrings;
```

Avoid Disinformation

```
int O1 = 0;
int l = 0;
int O = 0;
int a = l;


if (O == 5)
    a = O1;
else
    l = 01;
```

Make Meaningful Distinctions

```
public static void copyChars(char a1[], char a2[]) {
    for (int i = 0; i < a1.length; i++) {
        a2[i] = a1[i];
    }
}
```

Make Meaningful Distinctions

```
private aBall;
private theBall;

getActiveAccount();
getActiveAccounts();
getActiveAccountInfo();

String cust;
String customer;

class ProductInfo { … }
class ProductData { … }

String nameString;
```

Use Pronounceable Names

FLDSMDFR

Flint Lockwood's Diatonic Super Mutating Dynamic Food Replicator

Use Pronounceable Names

```
class DtaRcrd102 {
    private Date genymdhms;
    private Date modymdhms;
    private final String pszqint = "102";
    /* ... */
};

//to

class Customer {
    private Date generationTimestamp;
    private Date modificationTimestamp;;
    private final String recordId = "102";
    /* ... */
};
```

Use Pronounceable Names

```
for (int j = 0; j < 34; j++) {
    s += (t[j] * 4) / 5;
}

//to

int realDaysPerIdealDay = 4;
int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for (int j = 0; j < NUMBER_OF_TASKS; j++) {

    int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
    int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);
    sum += realTaskWeeks;
}
```

Avoid Encodings - Hungarian Notation, Member Prefixes

```
BOOL            m_bFormLevelEdit;       // Tracks whether or not Form level relationships should be validated
BOOL            m_bRowLevelEdit;        // Tracks whether or not Row level relationships should be validated
BOOL            m_bPerformingGet;       // Tracks whether or not the BO is currently performing a Get
BOOL            m_bPerformingSet;       // Tracks whether or not the BO is currently performing a Set

                                        // Variables used for loggin actions performed on a Business Object
BOOL            m_bEnableLogging;   // Turns Logging On and Off
CNpsString          m_strBOName;        // Every BO has to have a Name, Makes it easier to keep track of
CNpsString          m_strDataSource;    // Tracks what type of UI is interfacing with the BO
CNpsString          m_strModuleCode;    // Tracks what module is associated with this Business Object
CNpsString          m_strBOType;        // Business Object Type specifier  (It is really up to the programmer a
int             m_nFormState;       // Used to track the Form State

                                    // Variables used for User Defined Fields
BOOL            m_bHasUDF;                  // Have any UDFs been setup in Admin for this BO (Document or Detai
BOOL            m_bHasUDFDocument;          // Have any UDFs been setup in Admin for this BO (Document level on
CNpsString          m_strUDFDocumentTypeID;     // holds the Document level Type ID for the current bo.
UDFStruct*      m_pUDFStruct;               // holds an array of BODTSs for each UDF group (0-1 Doc level, 0-n
int             m_nSizeOfUDF;               // size of the m_pUDFStruct
CUDFDataArray*  m_pUDFDocData;              // Variant Array of Document level UDF Data.
BOOL            m_bLoadingFilterLookup;
```

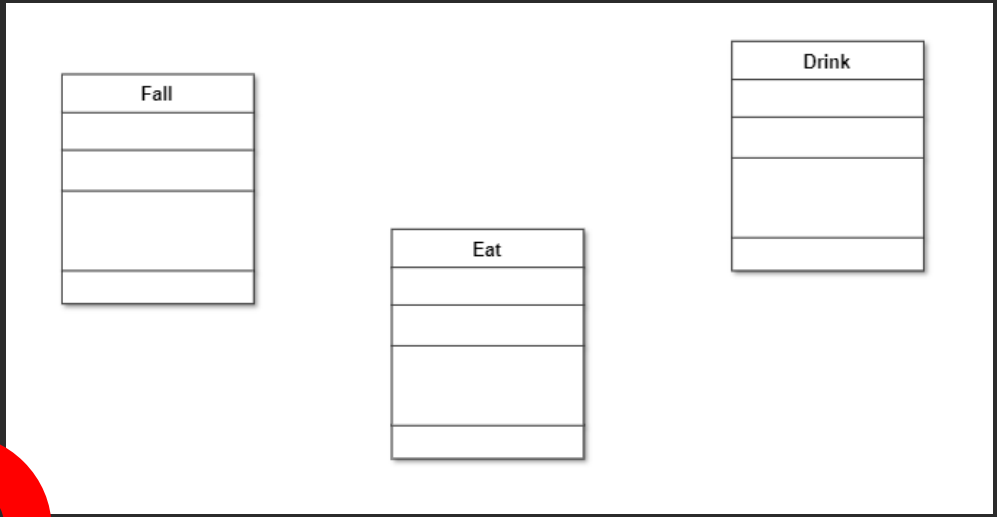
Use Pronounceable Names

```
public interface IShapeFactory {
}

public class ShapeFactory implements IShapeFactory{
}



//to

public interface ShapeFactory {
}

public class ShapeFactoryImp implements ShapeFactory{
}
```

Avoid Mental Mapping

```
for (int j = 0; j < NUMBER_OF_TASKS; j++) {

    int a = taskEstimate[j] * realDaysPerIdealDay;
    int b = (realdays / WORK_DAYS_PER_WEEK);
    sum += b;
}
```

Class Names



Method Names

```
Orange getCustomer(int airplane) {} !!!

void parse(int command) {}  ?
```

Don’t Be Cute

```
eatMyShorts()
abort()
```

Pick One Word per Concept

```
fetch()
retrieve()
get()
```

Use Solution Domain Names

AccountVisitor
JobQueue

Don’t Add Gratuitous Context

Address GSDAccountAddress;

“Gas Station Deluxe,”

“One difference between a smart programmer and a professional programmer is that the professional understands that clarity is king. Professionals use their powers for good and write code that others can understand”

“La diferencia entre un programador inteligente y un programador profesional es que este último sabe que la claridad es lo que importa. Los profesionales usan sus poderes para hacer el bien y escriben código que otros pueden entender”

Robert C. Martin