# Being a better programmer

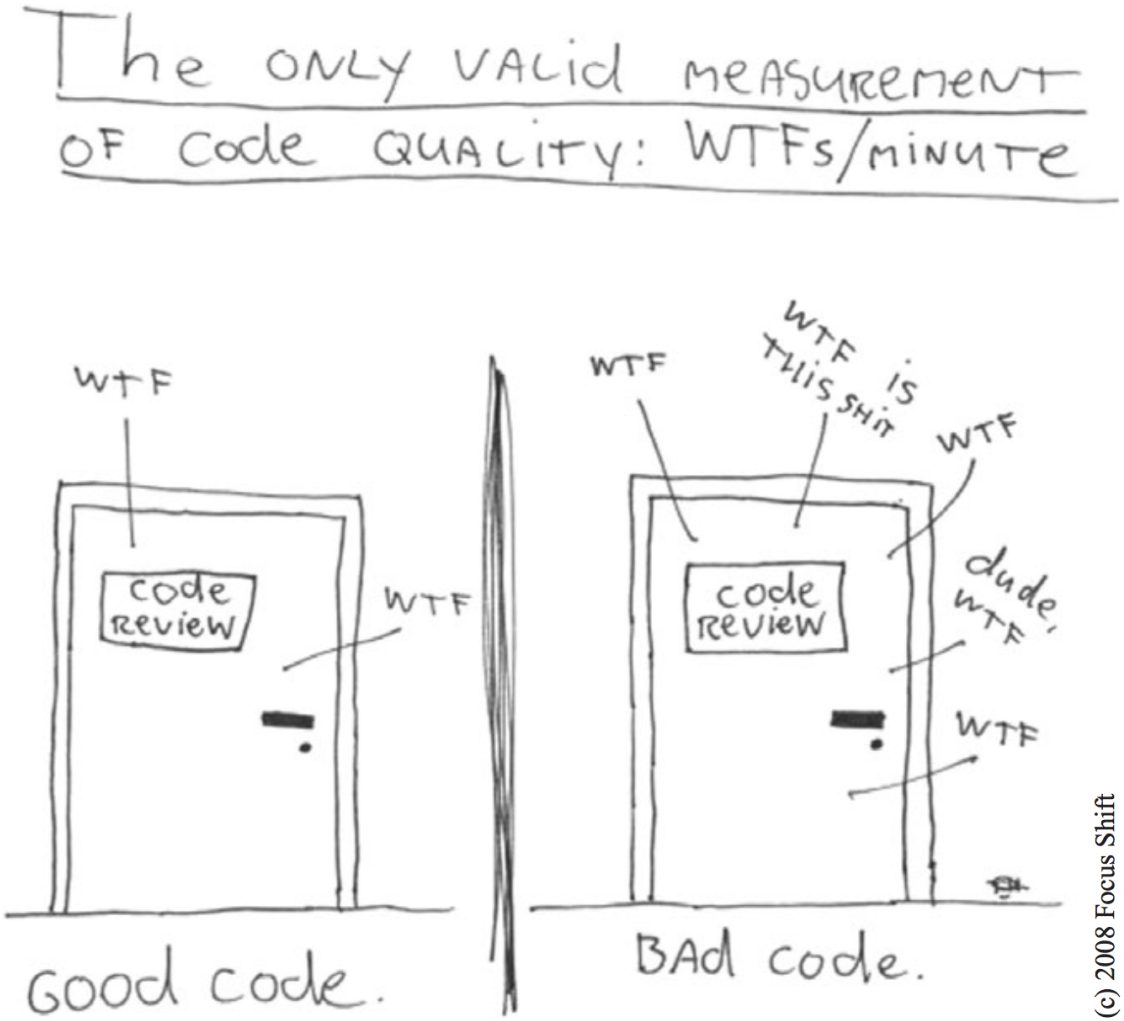
Writing Clean ~~Code~~ COBOL

Lean Bytes

*“Writing clean code is what you must do in order to call yourself a professional. There is no reasonable excuse for doing anything less than your best.”*

Robert C Martin

# Code Hell




Reproduced with the kind permission of Thom Holwerda.
http://www.osnews.com/story/19266/WTFs_m

# Code Hell Example

```
# get the count of rows in the database
# $x is the row count going forwards

my $arrayref = $sth->fetchall_arrayref;
$sth->finish;

my %label_hash;
foreach (@{$arrayref}) {
    $label_hash{$_->[0]} = $_->[1];
}

# next line gets the values but first sorts by the values in the hash
# not the keys themselves. Ref: Perl Cookbook P.144
my @values =  sort { $label_hash{$a} cmp $label_hash{$b} } keys %label_hash;

if (scalar @values eq 0) {

    return $query->b($APP_OPTIONS{'text_no_admins_available'});

} else {

    return $query->popup_menu(-name     => 'admin_id',
                              -values   => \@values,
                              -labels   => \%label_hash,
                              );
}
```

# Code Hell Example

```
# get the count of rows in the database
# $x is the row count going forwards

my $arrayref = $sth->fetchall_arrayref;
$sth->finish;

my %label_hash;
foreach (@{$arrayref}) {
    $label_hash{$_->[0]} = $_->[1];
}

# next line gets the values but first sorts by the values in the hash
# not the keys themselves. Ref: Perl Cookbook P.144
my @values =  sort { $label_hash{$a} cmp $label_hash{$b} } keys %label_hash;

if (scalar @values eq 0) {

    return $query->b($APP_OPTIONS{'text_no_admins_available'});

} else {

    return $query->popup_menu(-name      => 'admin_id',
                              -values    => \@values,
                              -labels    => \%label_hash,
                              );
}
```

Random old comment

Badly named variables

What is label_hash?
What's the point of this?

Useful comment?

What does this mean?

You’re not just a coder or a hacker
but a **professional**

so get out of the ***zone***
and **craft your code**

# Code for your fellow programmers

- Write **clean code** so that your fellow programmers will be able to **read** and **understand** it **easily**.
- Write **clean code** so that your fellow programmers will be able to **maintain** and **extend** it **easily**.
- Write **tests** for your clean code so that your fellow programmers will **know** what it does and can **change** it with **confidence**.

# Code for yourself

- Write clean code that **you will understand** when you come back to it **six months later**.
- Write clean code that **you feel proud of** and will still feel proud of **six months later**.
- Write **tests** for your clean code so that you can **extend** it or **refactor** it **easily**.
- Write clean code because you're a **professional** and you **love** your **craft**.

# Clean Code Principles

- Use MeaningfulNamesForYourVariables
- Write **small functions** that do **one thing only** and have **no side-effects**
- **Avoid excessive comments**, avoid comments
- **Abstract** your data into structures or objects
- Provide error handling, **raise exceptions**
- Write **unit tests** for your code
- **Write** your code, then **refactor** it; **refactor** it again

# Clean COBOL

# What is COBOL?

- **CO**mmon **B**usiness **O**rientated **L**anguage
- “Invented” by Grace Hopper, who was the inventor of FLOW-MATIC.
- Standardised between 1959 and 1960 by CODASYL (Pentagon).
- Designed to be platform and proprietor independent.
- Code should be readable by managers, business people and not just computer scientists.
- Originally an ANSI standard, now ISO with latest specification in 2014!

# COBOL Features:
# 500+ Reserved Words!

ABS, ACOS, ANNUITY, ASIN, ATAN, BYTE-LENGTH, CHAR, COMBINED-DATETIME,
CONCATENATE, COS, CURRENCY-SYMBOL, CURRENT-DATE, DATE-OF-INTEGER,
DATE-TO-YYYYMMDD, DAY-OF-INTEGER, DAY-TO-YYYYDDD, E, EXCEPTION-FILE,
EXCEPTION-LOCATION, EXCEPTION-STATEMENT, EXCEPTION-STATUS, EXP, EXP10,
FACTORIAL, FORMATTED-CURRENT-DATE, FORMATTED-DATE, FORMATTED-DATETIME,
FORMATTED-TIME, FRACTION-PART, HIGHEST-ALGEBRAIC, INTEGER,
INTEGER-OF-DATE, INTEGER-OF-DAY, INTEGER-OF-FORMATTED-DATE,
INTEGER-PART, LENGTH, LENGTH-AN, LOCALE-COMPARE, LOCALE-DATE,
LOCALE-TIME, LOCALE-TIME-FROM-SECONDS, LOG, LOG10, LOWER-CASE,
LOWEST-ALGEBRAIC, MAX, MEAN, MEDIAN, MIDRANGE, MIN, MOD,
MODULE-CALLER-ID, MODULE-DATE, MODULE-FORMATTED-DATE, MODULE-ID,
MODULE-PATH, MODULE-SOURCE, MODULE-TIME, MONETARY-DECIMAL-POINT,
MONETARY-THOUSANDS-SEPARATOR, NUMERIC-DECIMAL-POINT,
NUMERIC-THOUSANDS-SEPARATOR, NUMVAL, NUMVAL-C, NUMVAL-F, ORD, ORD-MAX,
ORD-MIN, PI, PRESENT-VALUE, RANDOM, RANGE, REM, REVERSE,
SECONDS-FROM-FORMATTED-TIME, SECONDS-PAST-MIDNIGHT, SIGN, SIN, SQRT,
STANDARD-DEVIATION, STORED-CHAR-LENGTH, SUBSTITUTE, SUBSTITUTE-CASE,
SUM, TAN, TEST-DATE-YYYYMMDD, TEST-DAY-YY
TEST-NUMVAL, TEST-NUMVAL-C, TEST-NUMVAL-
WHEN-COMPILED, YEAR-TO-YYYY IDENTIFICATIO
EVALUATEWHEN IS THEN IF END PROGRAM FUN

COBOL has some 500+ reserved words.

C in contrast has just 50.

Prolog has none!

# COBOL Features: Legibility

```
import  java.math.BigDecimal;
public  class  SalesTaxWithBigDecimal
{
  public static void main(java.lang.String[] args)
  {
    BigDecimal beforeTax     = BigDecimal.valueOf(12345, 2);
    BigDecimal salesTaxRate  = BigDecimal.valueOf(65, 3);
    BigDecimal ratePlusOne   =
salesTaxRate.add(BigDecimal.valueOf(1));
    BigDecimal afterTax    = beforeTax.multiply(ratePlusOne);
    afterTax  =  afterTax.setScale(2,  BigDecimal.ROUND_HALF_UP);
    System.out.println(  "After tax amount is "  + afterTax);
} }
```

```
identification division.
program-id.  SalesTax.
working-storage section.
01 beforeTax      picture 999V999 value 123.45.
01 salesTaxRate   picture V9999     value .065.
01 afterTax       picture 999.99.
procedure division.
Main.
      compute afterTax rounded = beforeTax + (beforeTax * salesTaxRate)
      display “After tax amount is “ afterTax.
```

# But doesn't COBOL suck?

- Well yes, perhaps it does.
- It's really, really old.
- There's some horrendous code out there. Lots of use of GO TO and other spaghetti code techniques.
- It predated structured programming, object oriented programming. A lot of code lacks sub-programs, functions or classes.
- But times have changed, and perhaps bad programmers are wont to blame their tools?

# Clean COBOL example

https://github.com/mikebharris/

https://github.com/OpenCobolIDE/

https://sourceforge.net/projects/open-cobol/

# Summary

- Become a better programmer, be a professional and craft your code.
- Read Bob Martin's Clean Code and The Clean Coder.  Read them again.
- Write tests.
- Bad code can be written in the latest and greatest trendy new Google language; good code can be written in an old dinosaur of a language.
- COBOL doesn't suck as much as you might have thought.

# Thank you.

07811 671 893

mike.harris@leanbytes.co.uk
http://leanbytes.co.uk
https://mbharris.co.uk
http://uk.linkedin.com/in/mbharris
https://github.com/mikebharris/



Lean Bytes