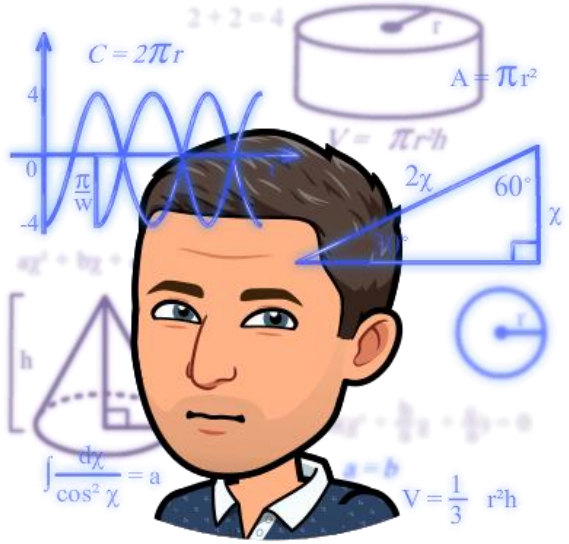
CLEAN CODE

BY YOAN THIRION

YOUR DEFINITION OF CLEAN CODE

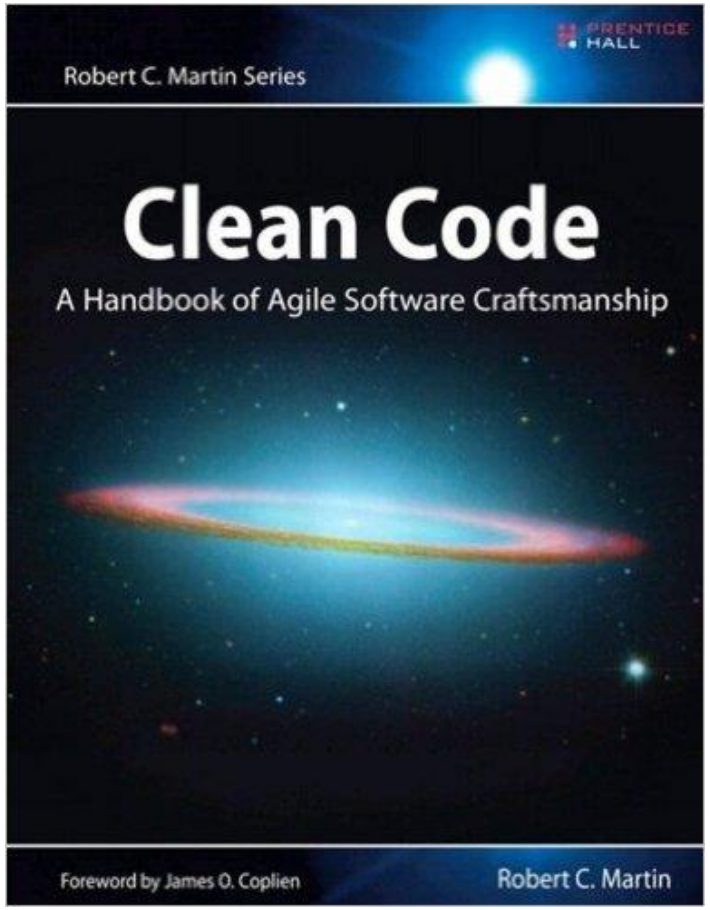
IMAGINE YOU OPEN THE MOST BEAUTIFUL CODE ON EARTH. IT IS PERFECT !!!
WHAT DO YOU SEE IN THIS CODE ? WHY IS IT SO PERFECT ?

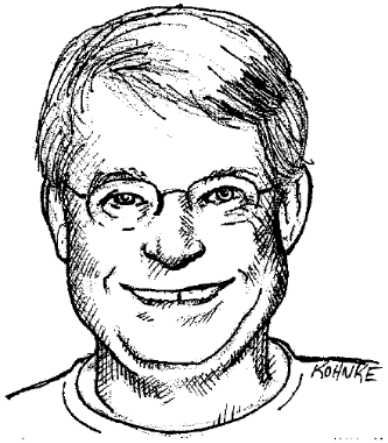


https://miro.com/welcomeonboard/EHZAQPO9XWTHCDTRGYPT1TSMOTCRN6OQRALNGV6TGMONO6BSWMCNTF4JOUEID3ROI

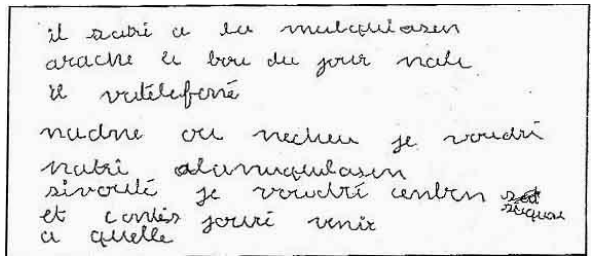
@YOT88

A WORD ON THE BOOK





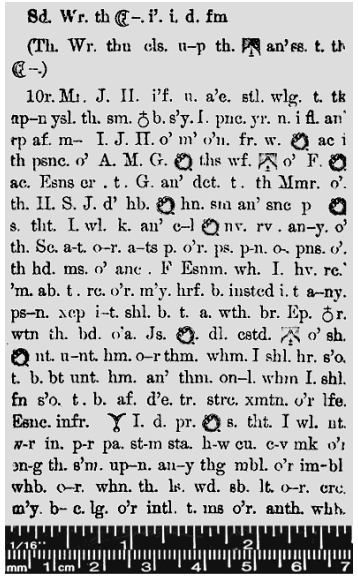
ROBERT C. MARTIN A.K.A
UNCLE BOB
PRENTICE HALL, 2009

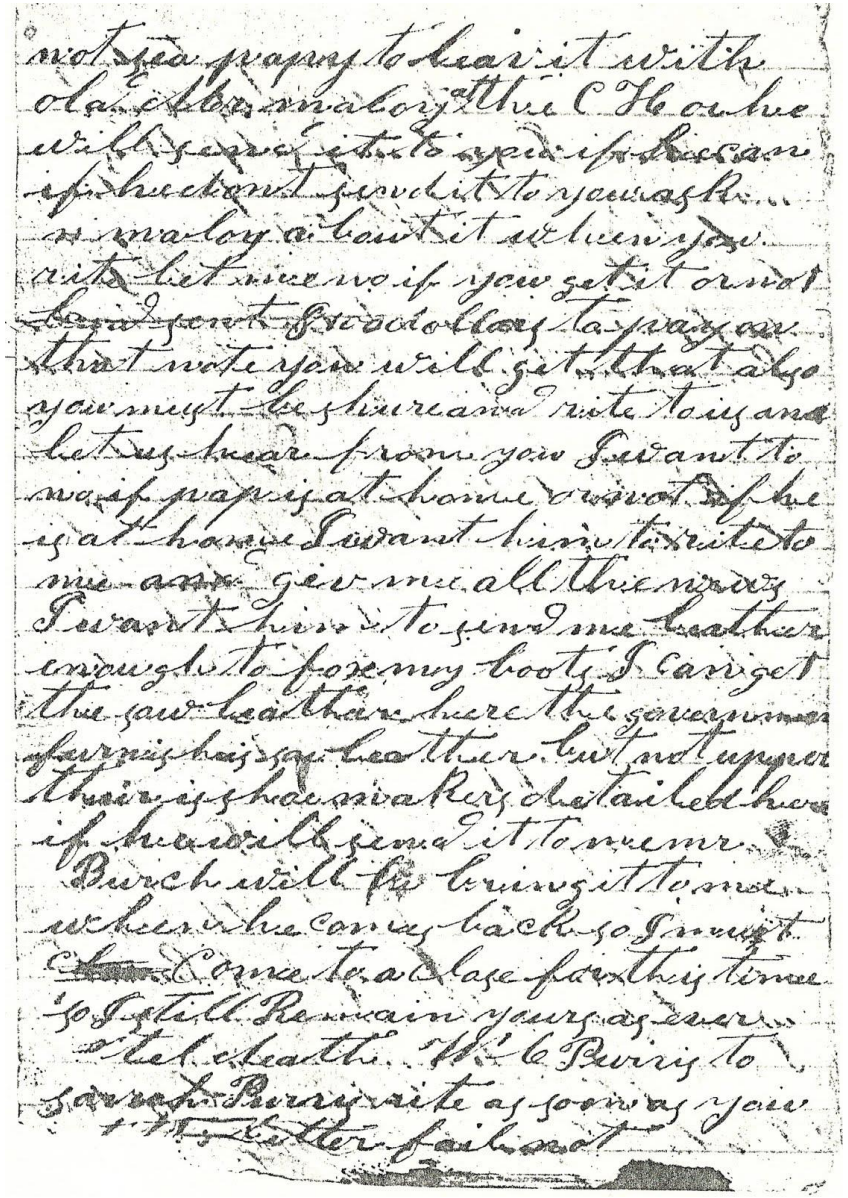


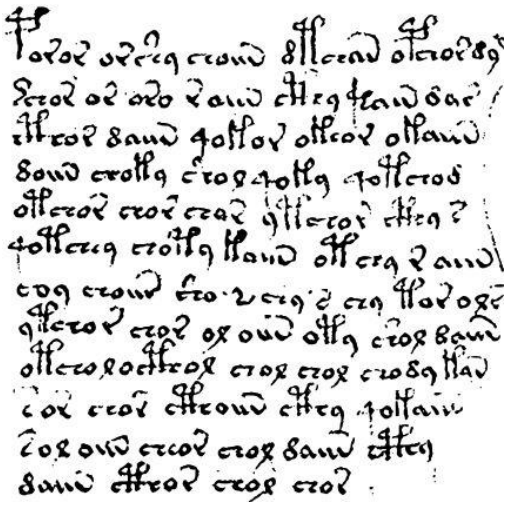
Sic s o s ik s hisssog s s hiks o ' s ik swa chigs hiss i o s is
o h s i ossa swo hsch ckigso s oo sb s hisso s a s s hikso
o

G ai siss ib sabo s h a igs ai ssos s ci s oss a sas w
chaiswi hsso higs osi i

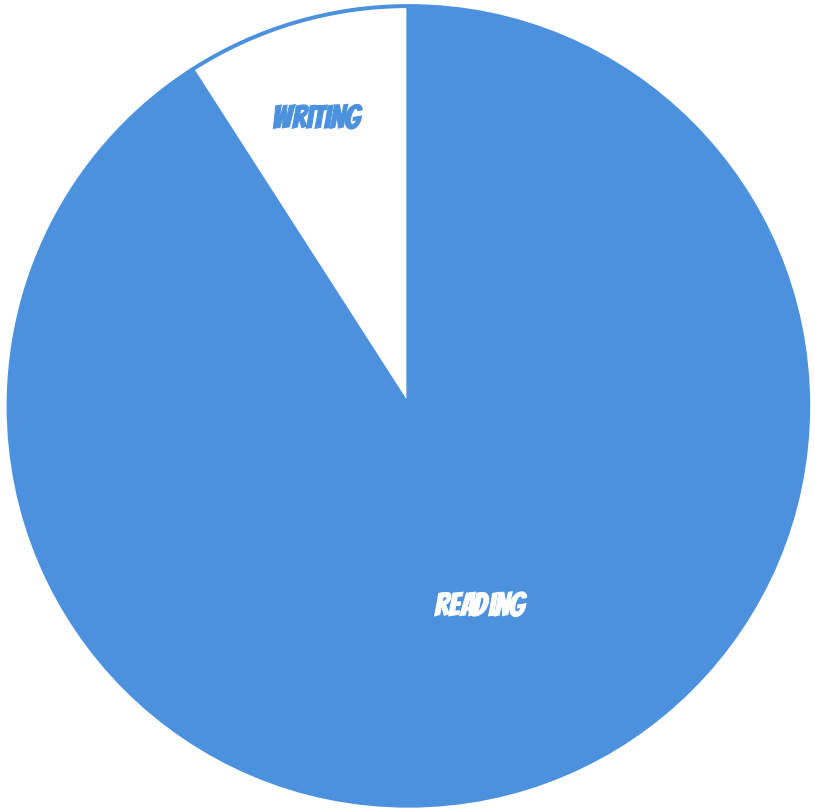
s o s as s ai s o sask swha 'ssgoigsosw hsMo 4Mika s s hik
i ss a swi hsaso aiascac sb igs o sa s h sso s o b
wi hsh sh a s o owigs h s oc s sas oso swoi ssa
s s ssh ss ss osb s oigsok sOb io s s o sh a hsiss ss os a
wi h sb ssh 'ssisgoo ss ii ssos h sGo igs h a







TIME SPENT BY PROGRAMMERS AT WORK



PROGRAMMERS = AUTHORS

OUR AUDIENCE = OTHER PROGRAMMERS

WHAT IS BAD CODE ?

- NOT EASY TO UNDERSTAND
- CLASSES ARE TIGHTLY COUPLED
- NOT EASY TO CHANGE
- DOES NOT COMMUNICATE THE INTENT
- SHOWS TOO MUCH OF THE INTERNALS (BAD ENCAPSULATION)

CODE SMELLS ARE WARNING SIGNS IN THE CODE

Shotgun Surgery

Switch statements

Uncommunicative Name

Too many arguments

Large class

Comments

Duplicated code

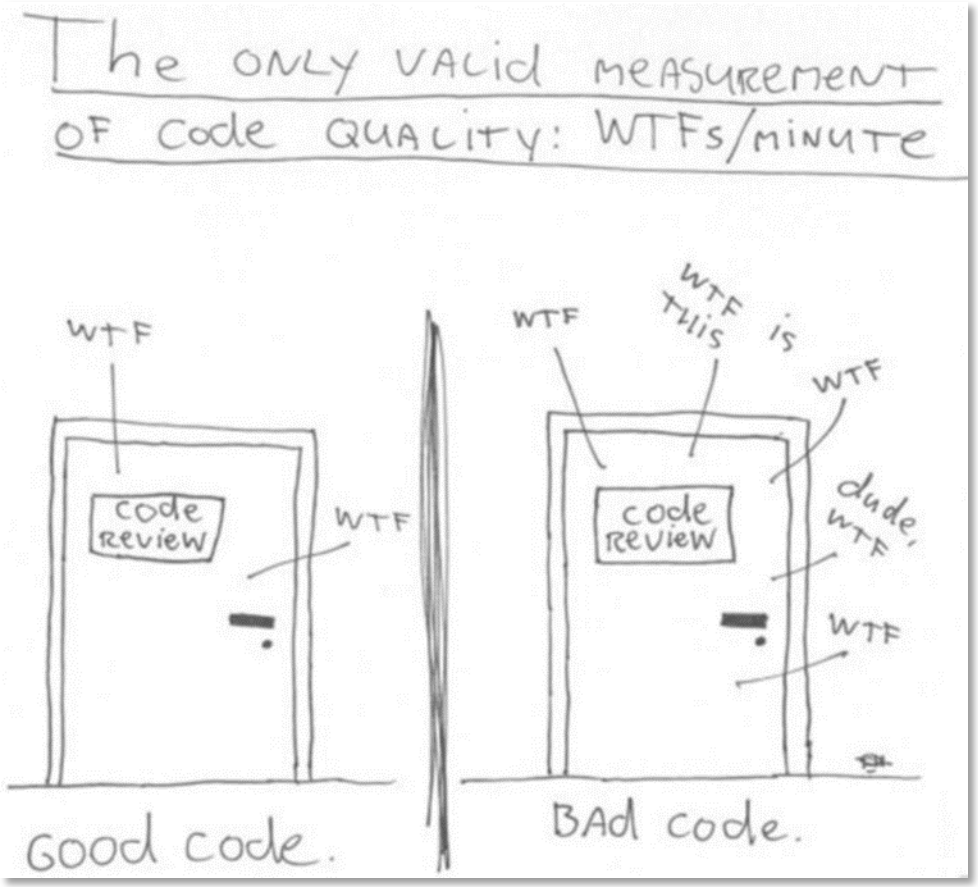
Dead Code

Inconsistent Names

Long methods

Conditional Complexity

"THE ONLY VALID MEASUREMENT OF CODE QUALITY IS THE NUMBER OF WTFS PER MINUTE" - ROBERT C. MARTIN



4 BIG TOPICS

NAMING
COMMENTS
FORMATTING
FUNCTIONS

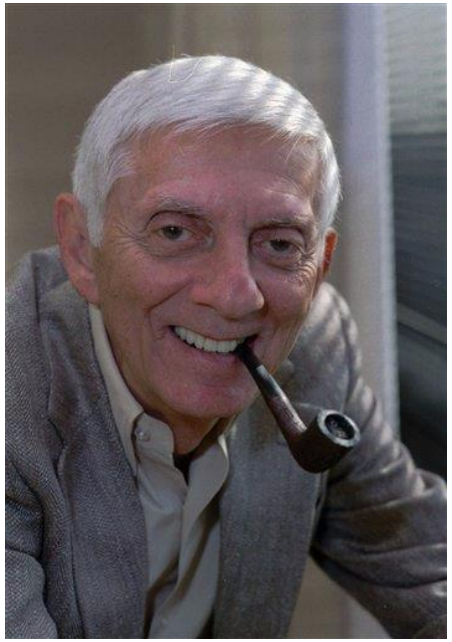
```
1 int d;
2 // elapsed time in days
3 int ds;
4 int dsm;
5 int faid;
```

NAMES MUST REVEAL YOUR INTENTIONS

```
1 int elapsedTimeInDays;
2 int daysSinceCreation;
3 int daysSinceModification;
4 int fileAgeInDays;
```

```
1 Customer[] customerList;
2 Table theTable;
```

AVOID DISINFORMATION

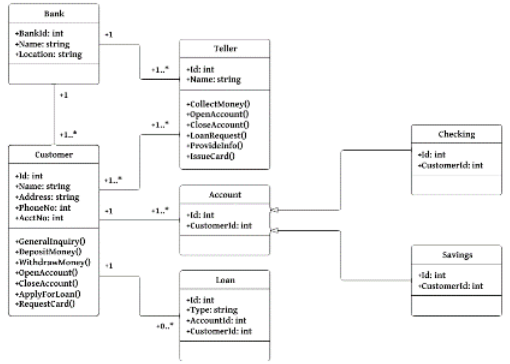


```
1 Customer[] customers;
2 Table customers;
```

```
1 string addressCity;
2 string addressHomeNumber;
3 string addressPostCode;
```

USE MEANINGFUL NAMES IN THEIR SELF CONTEXT

```
1 class Address
2 {
3 string city;
4 string homeNumber;
5 string postCode;
6 }
```



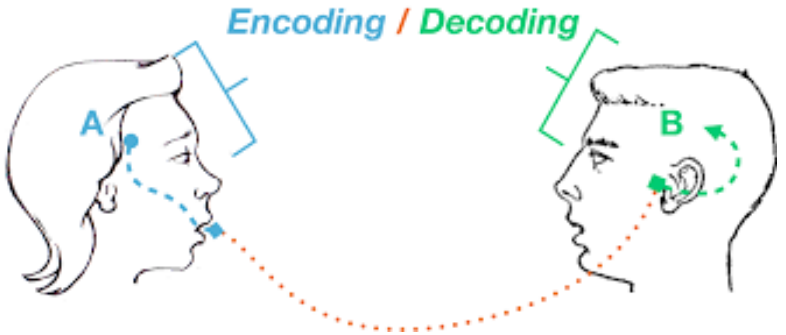
```
1 var theCustomersListWithAllCustomersIncludedWithoutFilter;
2 bool visibleStateCheckWhenCustomerAccessGranted;
```

GOOD NAMES LENGTH

```
1 var allCustomers;
2 bool isVisible;
```

```
1 private string m_strExePath;
```

AVOID ENCODINGS




```
1 private string executablePath;
```

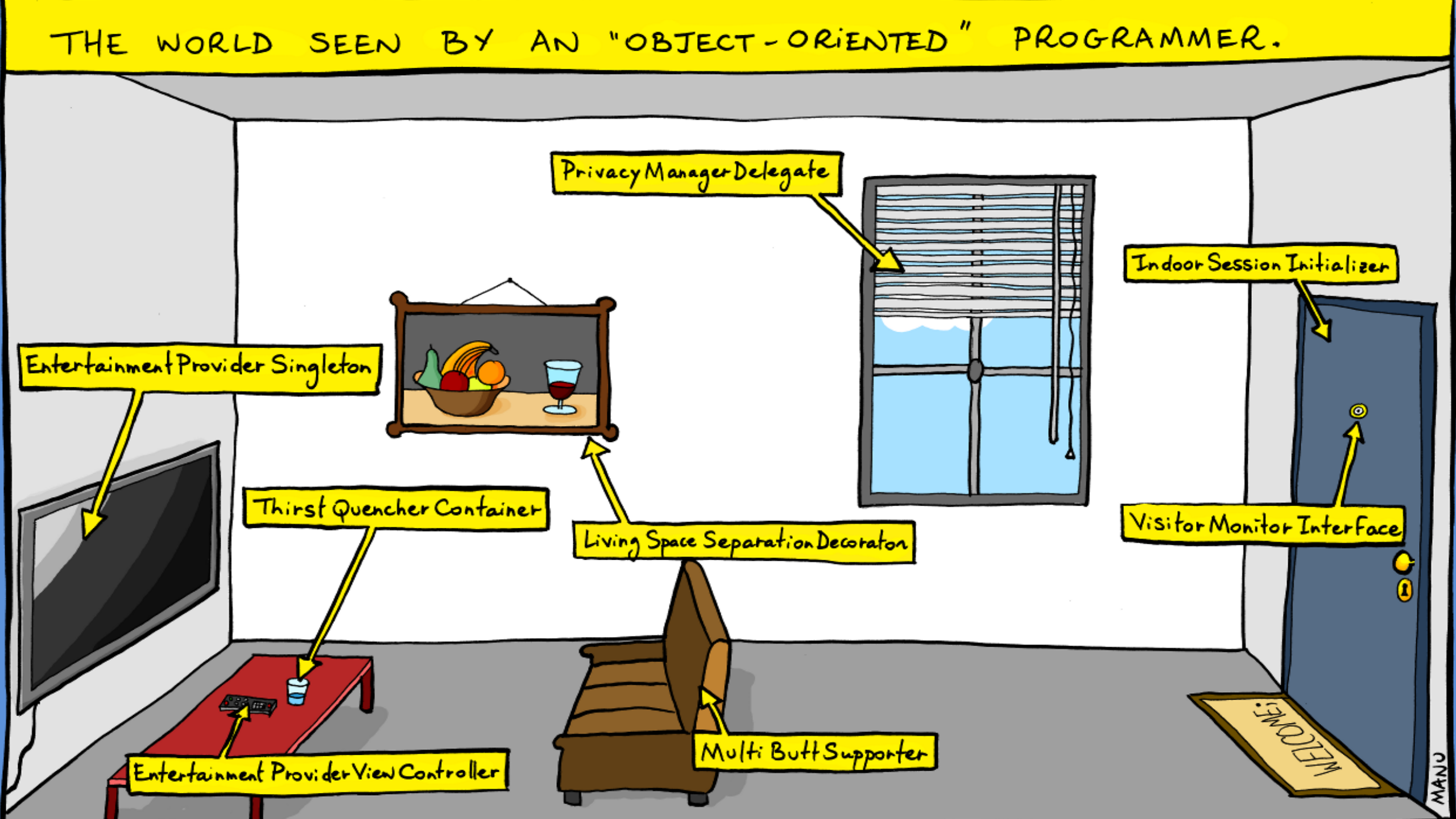
Class Names

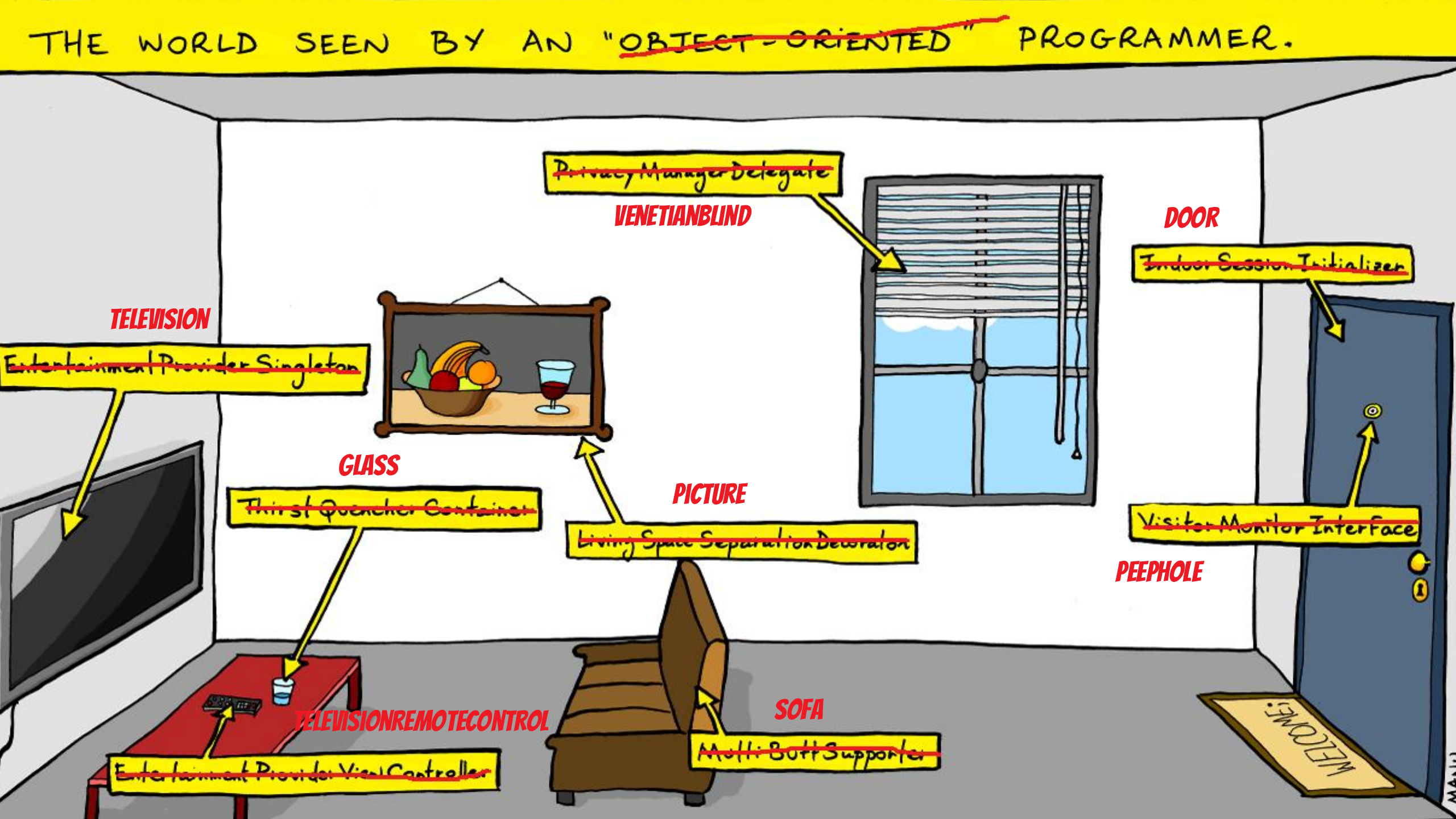
- Classes and objects should have noun or noun-phrase names
- A class name should not be a verb.

Java class names from Spring Framework :

- SimpleBeanFactoryAwareAspectInstanceFactory
- AbstractInterceptorDrivenBeanDefinitionDecorator
- AbstractSingletonProxyFactoryBean

THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.



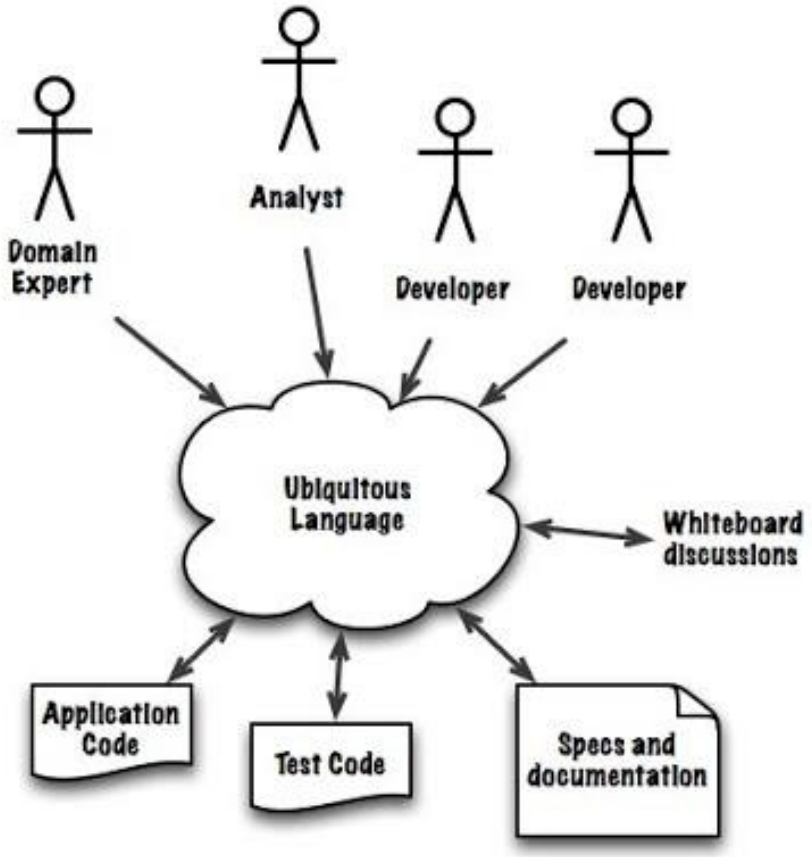

THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.
PrivacyManagerDelegate
VENETIANBLIND
DOOR
IndoorSessionInitializer
TELEVISION
EntertainmentProviderSingleton
GLASS
ThirstQuencherContainer
PICTURE
LivingSpaceSeparationDecorator
VisitorMonitorInterface
PEEPHOLE
TELEVISIONREMOTECONTROL
EntertainmentProviderViewController
SOFA
MultiButtSupporter
WELCOME.

USE AN UBIQUITOUS LANGUAGE

LANGUAGE USED :

BY ALL TEAM MEMBERS
BY ALL BUSINESS EXPERTS

IN THE SOURCE CODE AS WELL



WHAT ABOUT COMMENTS ?

"A COMMON FALLACY IS TO ASSUME AUTHORS OF INCOMPREHENSIBLE CODE WILL SOMEHOW BE ABLE TO EXPRESS THEMSELVES LUCIDLY AND CLEARLY IN COMMENTS."

–KEVLIN HENNEY

```
// The day of the year.
int dayOfYear = DateTime.Today.DayOfYear;
```

```
/// <summary>
/// Send email
/// </summary>
private void SendEmail()
{
  // ...
  // ...
}
```

REDUNDANT COMMENTS

```
/// <summary>
/// Utility method that returns when this.closed is true.
/// </summary>
/// <param name="timeoutInMilliseconds"></param>
public void WaitForClose(long timeoutInMilliseconds)
{
    if (!closed)
    {
        Wait(timeoutInMilliseconds);

        if (!closed)
        {
            throw new Exception("Sender could not be closed");
        }
    }
}
```

MISLEADING COMMENTS

CAN CONTAIN LIES

```
1  private void Persist<TEntity>(TEntity entity)
2  {
3      Save(entity);
4      //entity.Load();
5      //entity.Prepare();
6
7      /*if(state == true)
8      {
9
10     }*/
11 }
```

DON'T LEAVE COMMENTED OUT CODE IN YOUR CODEBASE

```
1  private void Persist<TEntity>(TEntity entity)
2  {
3      Save(entity);
4  }
```

```
/*
2016-10-09: Remove dead code (Yot)
2016-11-01: Improve performance (AM)
2016-11-03: Makes the method async (AM)
*/
private async Task Persist<TEntity>(TEntity entity)
{
    await Save(entity);
}
```

DON'T HAVE JOURNAL COMMENTS

USE YOUR SOURCE CONTROL INSTEAD



```
private async Task Persist<TEntity>(TEntity entity)
{
    await Save(entity);
}
```

```csharp
/// <summary>
/// This class describes a person.
/// </summary>
internal class Person
{
    /// <summary>
    /// Initializes a new Person instance.
    /// </summary>
    public Person()
    {
    }

    /// <summary>
    /// Gets or sets the person Identifier.
    /// </summary>
    public Guid Id { get; set; }

    /// <summary>
    /// Gets or sets the person Name.
    /// </summary>
    public string Name { get; set; }

    // ...
    // ...
    // ...
}
```

COMMENTS ARE NOISE

VALID USE OF COMMENTS

PUBLIC API COMMENTS

```
/// <summary>
/// Remove all items from this set. This clears the elements but not the underlying
/// buckets and slots array. Follow this call by TrimExcess to release these.
/// </summary>
public void Clear()
{
    if (_lastIndex > 0)
    {
        //..
    }
    _version++;
}
```

LEGAL COMMENTS

```
1 // Licensed to the .NET Foundation under one or more agreements.
2 // The .NET Foundation licenses this file to you under the MIT license.
3 // See the LICENSE file in the project root for more information.
```

TODO COMMENTS

```
1 //TODO : Refactor this method to make it async
2 private void Persist<TEntity>(TEntity entity)
3 {
4     Save(entity);
5 }
```

Task List

	Description
	TODO : Refactor this method to make it async

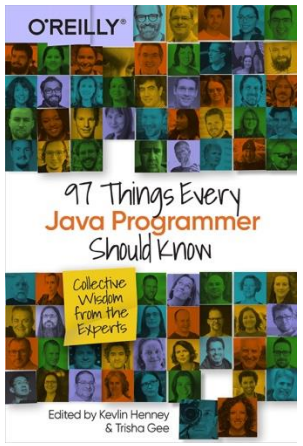
EXPLANATION OF INTENT

```
1  public virtual void OnDeserialization(Object sender)
2  {
3      if (_siInfo == null)
4      {
5          // It might be necessary to call OnDeserialization from a container if the
6          // container object also implements OnDeserialization. We can return immediately
7          // if this function is called twice. Note we set _siInfo to null at the end of this method.
8          return;
9      }
10     //..
11 }
```

ONE-SENTENCE DOCUMENTATION COMMENTS

- PETER HILTON

1. WRITE THE BEST CODE YOU CAN.
2. WRITE A ONE-SENTENCE DOCUMENTATION COMMENT FOR EVERY PUBLIC CLASS AND METHOD/FUNCTION.
3. REFACTOR THE CODE.
4. DELETE UNNECESSARY COMMENTS.
5. REWRITE BAD COMMENTS (BECAUSE ALL GOOD WRITING REQUIRES REWRITING).
6. ONLY ADD DETAIL WHERE NECESSARY.



COMMENTS ARE ANSWERING THE **WHY** QUESTIONS THAT YOU CAN'T ANSWER IN CODE

WHAT ABOUT FORMATTING ?

```
1  public class person
2  {
3      private const string CATEGORY = "P";
4      private string _name;
5
6      public void pay(decimal amount)
7      {
8        //...
9      }
10 }
11
12 public class Client
13 {
14     private const string category = "C";
15     private string name;
16
17     public void Pay(decimal amount)
18     {
19       //...
20     }
21 }
```

USE CONSISTENT CAPITALIZATION

```
1  public class Person
2  {
3      private const string CATEGORY = "P";
4      private string name;
5
6      public void Pay(decimal amount)
7      {
8          //...
9      }
10 }
11
12 public class Client
13 {
14     private const string CATEGORY = "C";
15     private string name;
16
17     public void Pay(decimal amount)
18     {
19         //...
20     }
21 }
```

USE CONSISTENT CAPITALIZATION

UNSUSTAINABLE SPACING

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over
Working software over comprehensiv
Customer collaboration over contra

Responding to change over following a plan

HOW MANY PROGRAMMERS LAY OUT THEIR CODE

COLUMN 80

HOW PEOPLE READ

```
public ResultType ArbitraryMethodName(FirstArgumentType firstArgument,
                                      SecondArgumentType secondArgument,
                                      ThirdArgumentType thirdArgument)
{
    LocalVariableType localVariable = Method(firstArgument,
                                             secondArgument);

    if (localVariable.isSomething(thirdArgument,
                                  SOME_SHOUTY_CONSTANT))
    {
        DoSomethingWith(localVariable);
    }
    return localVariable.getSomething();
}
```

```
1  public ResultType ArbitraryMethodName(
2      FirstArgumentType firstArgument,
3      SecondArgumentType secondArgument,
4      ThirdArgumentType thirdArgument)
5  {
6      LocalVariableType localVariable =
7          Method(firstArgument, secondArgument);
8
9      if (localVariable.isSomething(
10         thirdArgument, SOME_SHOUTY_CONSTANT))
11     {
12         DoSomethingWith(localVariable);
13     }
14     return localVariable.getSomething();
15 }
```

@YOI88

INDENTATION CAN REVEAL THINGS

```
1  public void Play()
2  {
3      var players = new List<Player>();
4
       foreach(var player in players)
       {
           foreach(var game in player.Games)
           {
               foreach(var bet in game.Bets)
               {
                   foreach(var sport in bet.Sports)
                   {

                   }
               }
           }
       }
18 }
```

CYCLOMATIC COMPLEXITY :

METRIC TO INDICATE THE COMPLEXITY OF A PIECE OF CODE
MEASURE THE NUMBER OF INDEPENDENT PATHS THROUGH OUR CODE

FUNCTIONS

```
1  private string ArabicNumeralToRoman(int num)
2  {
3      string val = "";
4
5      while (num >= 1000)
6      {
7          val += "M";
8          num -= 1000;
9      }
10     if (num >= 900)
11     {
12         val += "CM";
13         num -= 900;
14     }
15     while (num >= 500)
16     {
17         val += "D";
18         num -= 500;
19     }
20     if (num >= 400)
21     {
22         val += "CD";
23         num -= 400;
24     }
25     while (num >= 100)
26     {
27         val += "C";
28         num -= 100;
29     }
30     if (num >= 90)
31     {
32         val += "XC";
33         num -= (0;
34     }
```

FUNCTIONS SHOULD BE SMALL

(NO LONGER THAN ABOUT 6 OR SO LINES LONG)

```
public void RegisterUser(string email, string name)
{
    var regularExpression = new Regex(@"/^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$/i;");

    if(!regularExpression.IsMatch(email))
    {
        throw new InvalidOperationException("email invalid");
    }

    var user = new User(email, name);
    Register(user);
}
```

FUNCTIONS SHOULD DO ONLY ONE THING


```
public void RegisterUser(string email, string name)
{
    CheckEmail(email);
    var user = new User(email, name);
    Register(user);
}

private void CheckEmail(string email)
{
    var regularExpression = new Regex(@"/^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$/i;");

    if (!regularExpression.IsMatch(email))
    {
        throw new InvalidOperationException("email invalid");
    }
}
```

```
1 public double Convert(double value)
2 {
3     return (value - 32) * 5 / 9;
4 }
```

USE DESCRIPTIVE NAMES

```
1 public double ConvertToCelcius(double fahrenheit)
2 {
3     return (fahrenheit - 32) * 5 / 9;
4 }
```

FUNCTION ARGUMENTS

A FUNCTION SHOULD HAVE PREFERABLY NO ARGUMENTS (NILADIC).
OTHERWISE, PREFERABLY ONE (MONADIC),
TWO IS OKAY (DYADIC),
THREE (TRIADIC) SHOULD BE AVOIDED WHERE POSSIBLE,
AND YOU SHOULD NEVER HAVE FOUR OR MORE (POLYADIC).

Monadic functions

3 valid forms :

```
boolean FileExists("MyFile")
InputStream FileOpen("MyFile")
void onPasswordAttemptsFailed(int attempts)
```

Those taking "flag" arguments, such as

```
void Render(true/false)
```

Should be split in 2 functions that take no argument

DIADIC FUNCTIONS

THERE ARE TIMES WHEN TWO ARGUMENTS ARE CONCEPTUALLY IMPORTANT, SUCH AS

```
AddNewPoint(int x, int y)
```

> 2 ARGUMENTS : YOUR FUNCTION IS TRYING TO DO TOO MUCH ?

COMMAND/QUERY SEPARATION

```
1  private string CalculateRental()
2  {
3      string result = "";
4
5      foreach (var rental in this.rentals)
6      {
7          var thisAmount = rental.LineAmount();
8          this.FrequentRenterPoints += rental.CalculateFrecuentPoints();
9          this.AmountOwed = this.AmountOwed + thisAmount;
10
11         result += FormatLine(rental, thisAmount);
12     }
13
14     return result;
15 }
```

COMMAND/QUERY SEPARATION

A FUNCTION SHOULD EITHER :

DO SOMETHING : CHANGE THE STATE OF AN OBJECT – A COMMAND

RETURN SOMETHING : RETURN SOME INFORMATION ABOUT THAT OBJECT – A QUERY

NOT DO BOTH

Conditional that performs various actions depending on object type or properties.

```
class Bird {
  //...
  double getSpeed() {
    switch (type) {
      case EUROPEAN:
        return getBaseSpeed();
      case AFRICAN:
        return getBaseSpeed() - getLoadFactor() * numberOfCoconuts;
      case NORWEGIAN_BLUE:
        return (isNailed) ? 0 : getBaseSpeed(voltage);
    }
    throw new RuntimeException("Should be unreachable");
  }
}
```

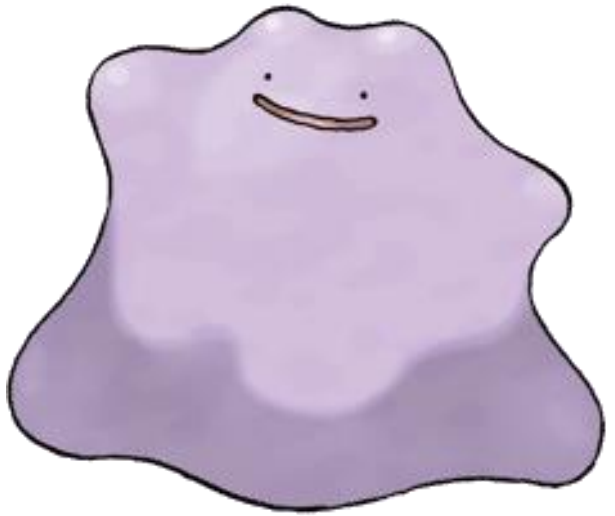
TELL-DON'T-ASK PRINCIPLE

REPLACE CONDITIONAL WITH POLYMORPHISM

```
abstract class Bird {
  //...
  abstract double getSpeed();
}

class European extends Bird {
  double getSpeed() {
    return getBaseSpeed();
  }
}
class African extends Bird {
  double getSpeed() {
    return getBaseSpeed() - getLoadFactor() * numberOfCoconuts;
  }
}
class NorwegianBlue extends Bird {
  double getSpeed() {
    return (isNailed) ? 0 : getBaseSpeed(voltage);
  }
}

// Somewhere in client code
speed = bird.getSpeed();
```



PREFER EXCEPTIONS TO RETURNING ERROR CODES

Before

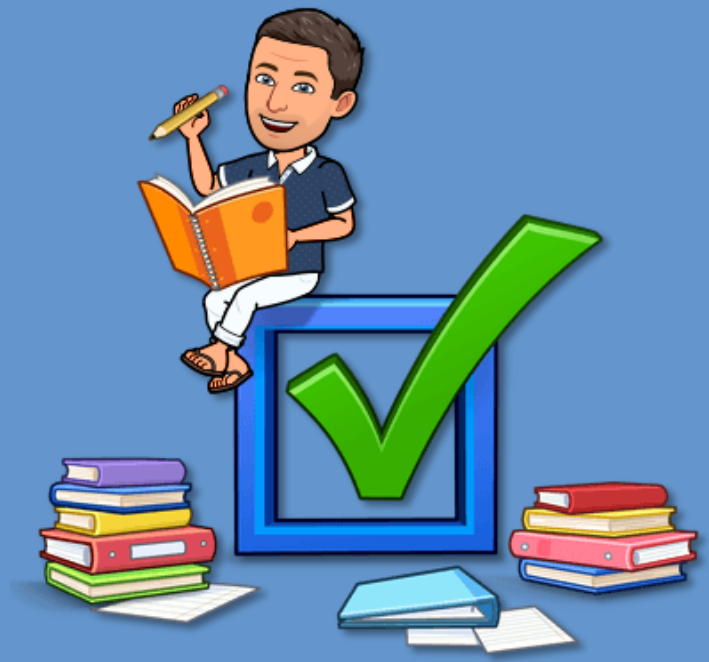
```
int Withdraw(int amount)
{
  if (amount > _balance)
  {
    return -1;
  }
  else
  {
    balance -= amount;
    return 0;
  }
}
```

After

```
void Withdraw(int amount)
{
  if (amount > _balance)
  {
    throw new BalanceException();
  }
  balance -= amount;
}
```

ACRONYMS

PLEASE



Clean Code Acronyms

Cheat Sheet

KISS

Keep It Simple and Stupid

simplicity should be a key goal in design, and that unnecessary complexity should be avoided

YAGNI

You Ain't Gonna Need It

programmers should not add functionality until it is necessary

always implement things when you need them, never when you just foresee that you need them

DRY

DRY

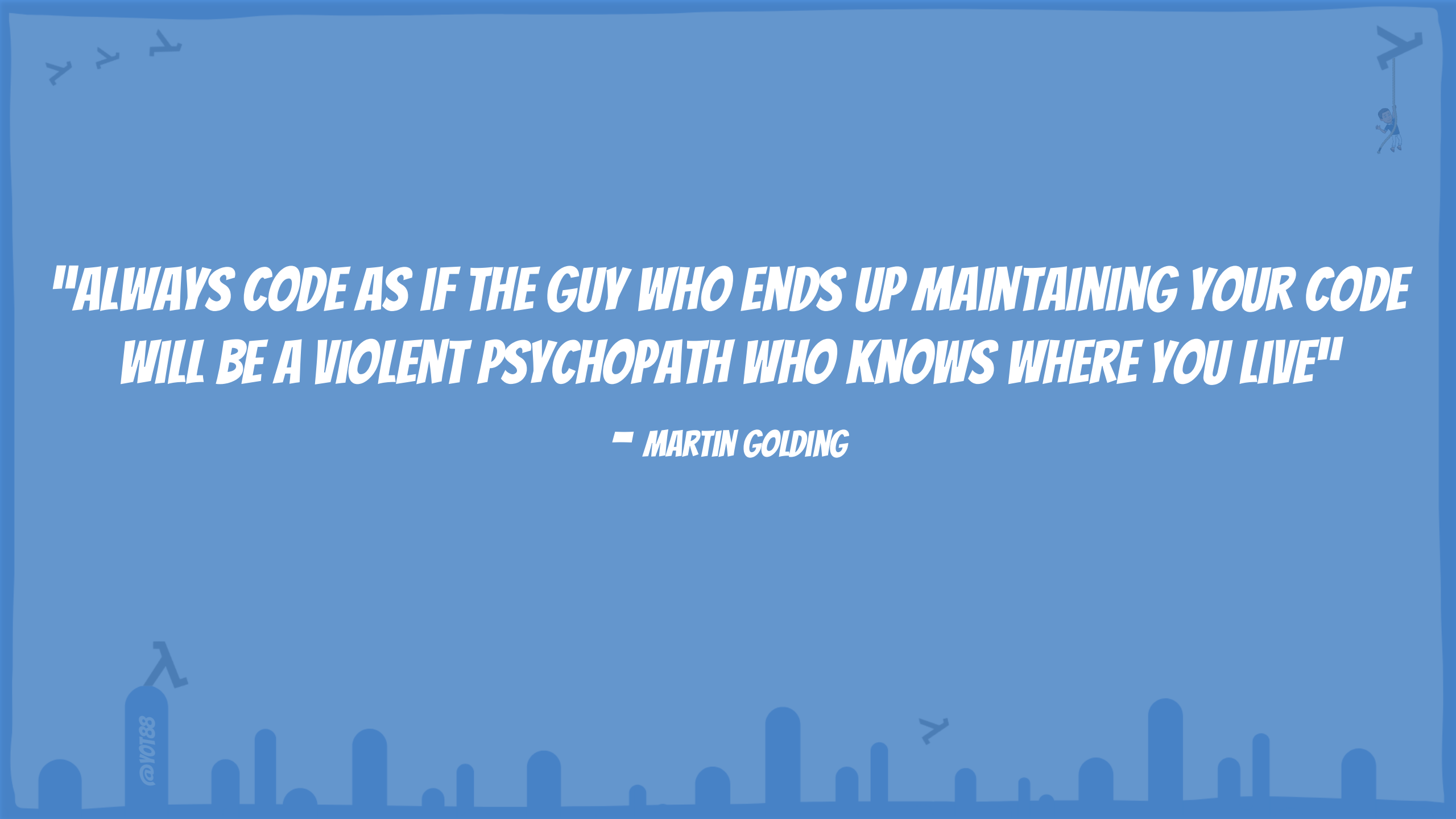
don't repeat yourself

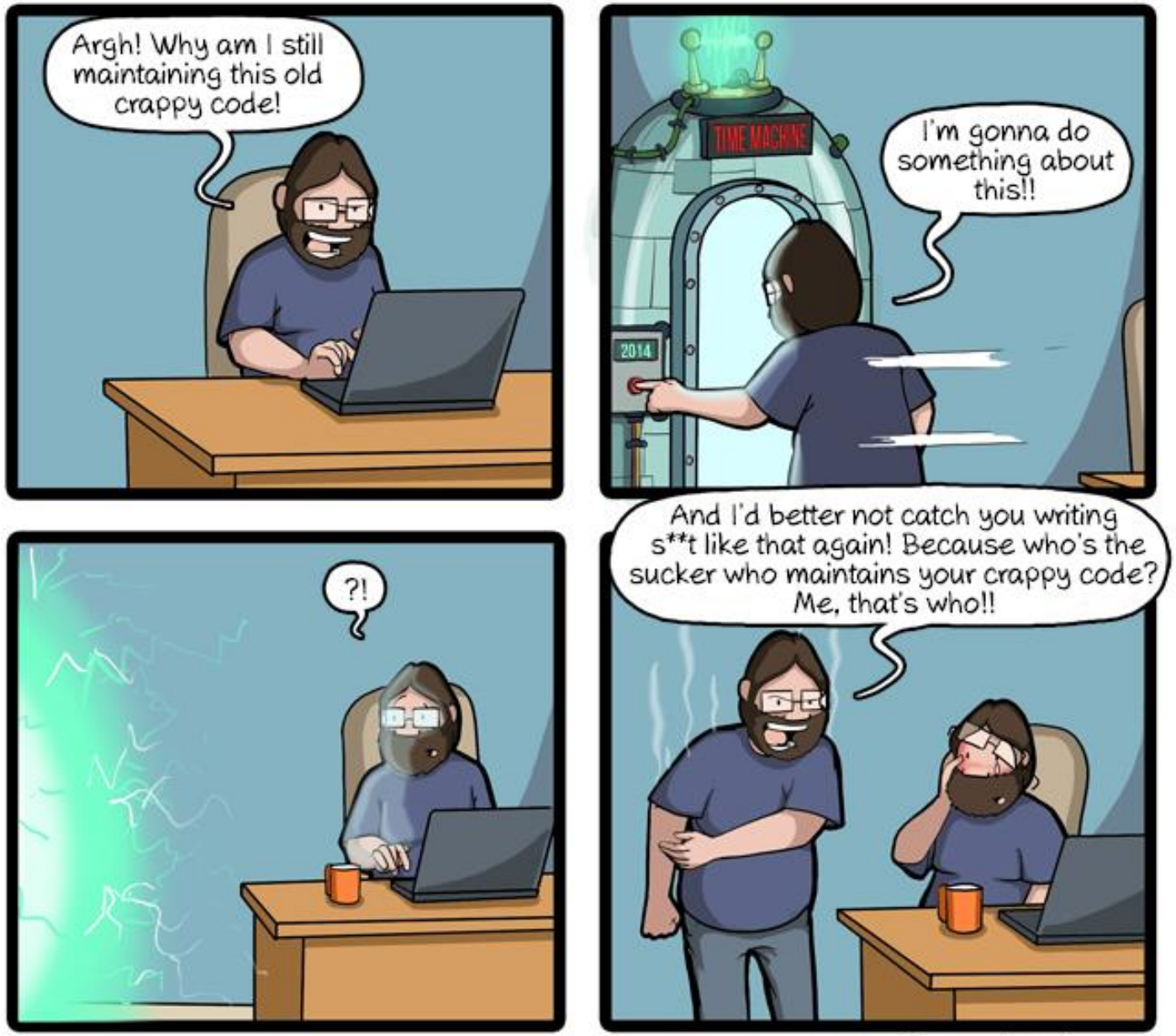
reducing repetition of information of all kinds

DIE

Duplication Is Evil

every piece of knowledge must have a single, unambiguous, representation within a system

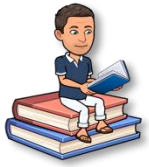

"ALWAYS CODE AS IF THE GUY WHO ENDS UP MAINTAINING YOUR CODE WILL BE A VIOLENT PSYCHOPATH WHO KNOWS WHERE YOU LIVE"
- MARTIN GOLDING


Argh! Why am I still maintaining this old crappy code!
TIME MACHINE
2014
I'm gonna do something about this!!
?!
And I'd better not catch you writing s**t like that again! Because who's the sucker who maintains your crappy code? Me, that's who!!
CommitStrip.com

4 SQUARE FEEDBACK

CONNECTIONS : MY FEELING ABOUT WHAT I HAVE LEARNED ARE...

LEARNED : THE MOST IMPORTANT CONCEPTS I LEARNED ARE...



CONCRETE PRACTICE : WHAT I PLAN TO DO WITH WHAT I LEARNED...

CONCLUSIONS : A FINAL COMMENT, SUGGESTION, OR QUESTION I STILL HAVE IS