OBJECT CALISTHENICS

Your code sucks, let's fix it!

OBJECT

Related to **O**bject **O**riented **P**rogramming.

CALISTHENICS

Term derived from greek, *exercise*, under the context of gymnastics.

OBJECT CALISTHENICS

Jeff Bay, in **The ThoughtWorks Anthology**, lists **9 rules** to writing better Object Oriented code.

MOTIVATION

Readable Code, Comprehensible, Testable, Maintainable.

1. ONLY ONE LEVEL OF INDENTATION PER METHOD

YAY!

```
class Board {
    public String board() {
        StringBuilder buf = new StringBuilder();

        // 0
        for (int i = 0; i < 10; i++) {
            // 1
            for (int j = 0; j < 10; j++) {
                // 2
                buf.append(data[i][j]);
            }
            buf.append("\n");
        }

        return buf.toString();
    }
}
```

EXTRACT METHOD

```java
public String board() {
    StringBuilder buf = new StringBuilder();

    collectRows(buf);

    return buf.toString();
}
```

```java
private void collectRows(StringBuilder buf) {
    for (int i = 0; i < 10; i++) {
        collectRow(buf, i);
    }
}

private void collectRow(StringBuilder buf, int row) {
    for (int i = 0; i < 10; i++) {
        buf.append(data[row][i]);
    }
    buf.append("\n");
}
```

http://refactoring.com/catalog/extractMethod.html

2. DON'T USE THE ELSE KEYWORD

EXAMPLE

```
public void login(String username, String password) {
    if (userRepository.isValid(username, password)) {
        redirect('homepage');
    } else {
        addFlash('error', 'Bad credentials');

        redirect('login');
    }
}
```

EARLY RETURN

```
public void login(String username, String password) {
    if (!userRepository.isValid(username, password)) {
        addFlash('error', 'Bad credentials');

        return redirect('login');
    }

    redirect('homepage');
}
```

VARIABLE

```
public void login(String username, String password) {
    String redirectRoute = 'homepage';

    if (!userRepository.isValid(username, password)) {
        addFlash('error', 'Bad credentials');
        redirectRoute = 'login';
    }

    redirect(redirectRoute);
}
```

BUT ALSO

- Polymorphism
- Null Object Pattern
- Strategy Pattern
- State Pattern

3. WRAP ALL PRIMITIVES AND STRINGS

Encapsulate all the primitives within objects.

http://c2.com/cgi/wiki?PrimitiveObsession

If the variable of primitive type has a **behavior**,
it **MUST** be encapsulated.

It is especially *true* for **Domain Driven Design**.

4. FIRST CLASS COLLECTIONS

Any class that contains a collection should contain no other member variables.

Each **collection** gets wrapped **in its own class**, so now behaviors related to the collection have a home.

(e.g. filter methods, applying a rule to each element)

5. ONE DOT PER LINE

LAW OF DEMETER

Only talk to your **immediate friends**, don't talk to strangers.

EXAMPLE

```java
class Location {                    class Piece {
    public Piece current;               public String representation;
}                                   }

class Board {
    public String boardRepresentation() {
        StringBuilder buf = new StringBuilder();

        for (Location loc : squares()) {
            buf.append(loc.current.representation.substring(0, 1));
        }

        return buf.toString();
    }
}
```

TALK TO YOUR FRIENDS!

```java
class Location {
    private Piece current;

    public void addTo(StringBuilder buf) { current.addTo(buf); }
}

class Piece {
    private String representation;

    public String character() {
        return representation.substring(0, 1);
    }

    public void addTo(StringBuilder buf) {
        buf.append(character());
    }
}
```

TALK TO YOUR FRIENDS!

```java
// Before:
// buf.append(loc.current.representation.substring(0, 1));

// After:
class Board {
    public String boardRepresentation() {
        StringBuilder buf = new StringBuilder();

        for (Location location : squares()) {
            location.addTo(buf);
        }

        return buf.toString();
    }
}
```

6. DON'T ABBREVIATE

WHY DO YOU WANT TO ABBREVIATE?

WRITE THE SAME NAME OVER AND OVER AGAIN?

Then, your method is reused multiple times.

Looks like **code duplication**.

METHOD NAME TOO LONG?

Maybe your class has multiple responsabilities.

Violation of the **Single Responsibility Principle**.

> *If you can't find a decent name for a class or a method, something is probably wrong with your conception. Rethink!*

Me — 2 hours ago

http://williamdurand.fr/2012/01/24/designing-a-software-by-naming-things/

7. KEEP ALL ENTITIES SMALL

No class over **50 lines** and no package over 10 files.

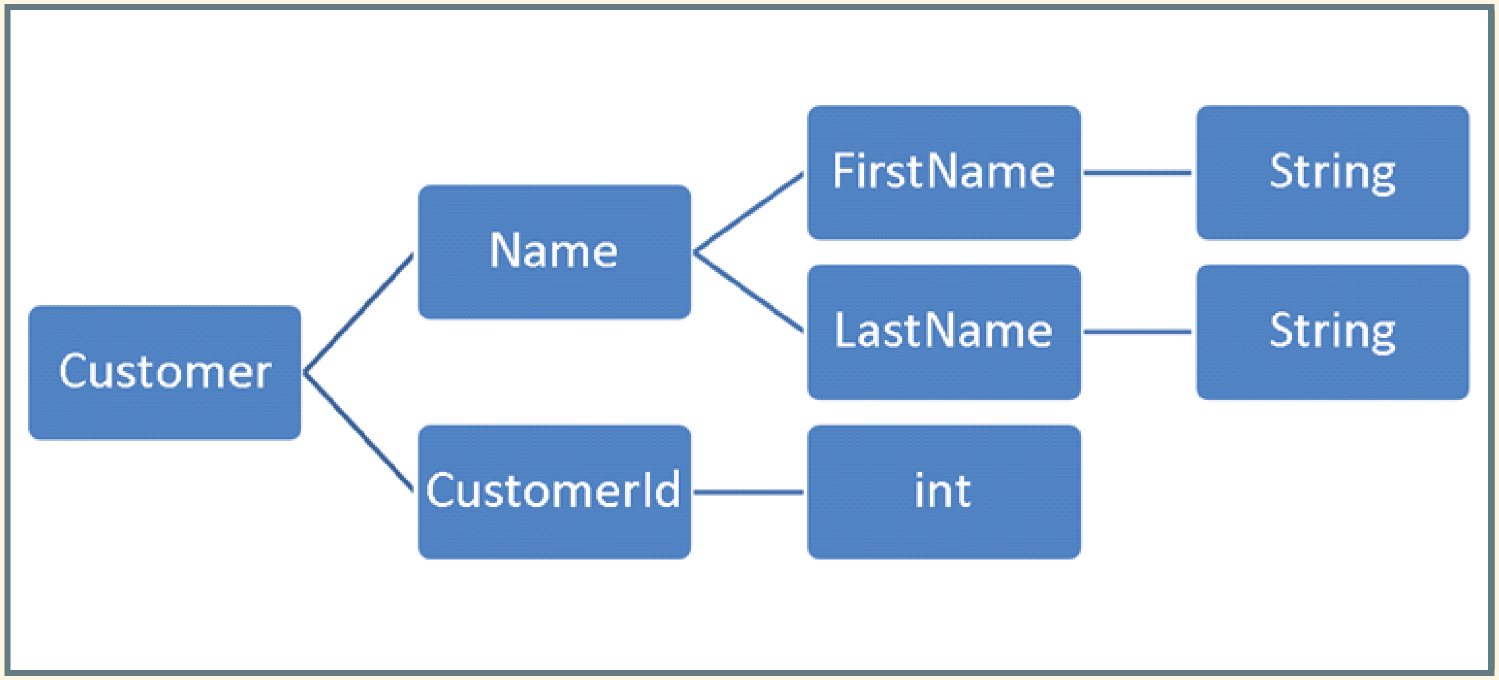
8. NO CLASSES WITH MORE THAN TWO INSTANCE VARIABLES

YES, I KNOW!

WHY?

Low cohesion, better encapsulation.

HOW?



https://github.com/TheLadders/object-calisthenics#rule-8-no-classes-with-more...

9. NO GETTERS/SETTERS/PROPERTIES

Tell, don't ask.

http://pragprog.com/articles/tell-dont-ask
http://c2.com/cgi/wiki?TellDontAsk

Getters/Setters violate the **Open/Closed Principle**.

GETTERS/SETTERS ARE EVIL

```
// Game
private int score;

public void setScore(int score) {
    this.score = score;
}

public int getScore() {
    return score;
}

// Usage
game.setScore(game.getScore() + ENEMY_DESTROYED_SCORE);
```

```
// Game
public void addScore(int delta) {
    score += delta;
}

// Usage
game.addScore(ENEMY_DESTROYED_SCORE);
```

http://stackoverflow.com/questions/565095/are-getters-and-setters-evil
http://whitewashing.de/2012/08/22/building_an_object_model__no_setters_allowed.html

RECAP’

1. Only One Level Of Indentation Per Method
2. Don't Use The ELSE Keyword
3. Wrap All Primitives And Strings
4. First Class Collections
5. One Dot Per Line
6. Don't Abbreviate
7. Keep All Entities Small
8. No Classes With More Than Two Instance Variables
9. No Getters/Setters/Properties

THANK YOU, QUESTIONS?

williamdurand.fr
github.com/willdurand
twitter.com/couac