# Code is for Humans

Jeremy Clark
jeremybytes.com
github.com/jeremybytes

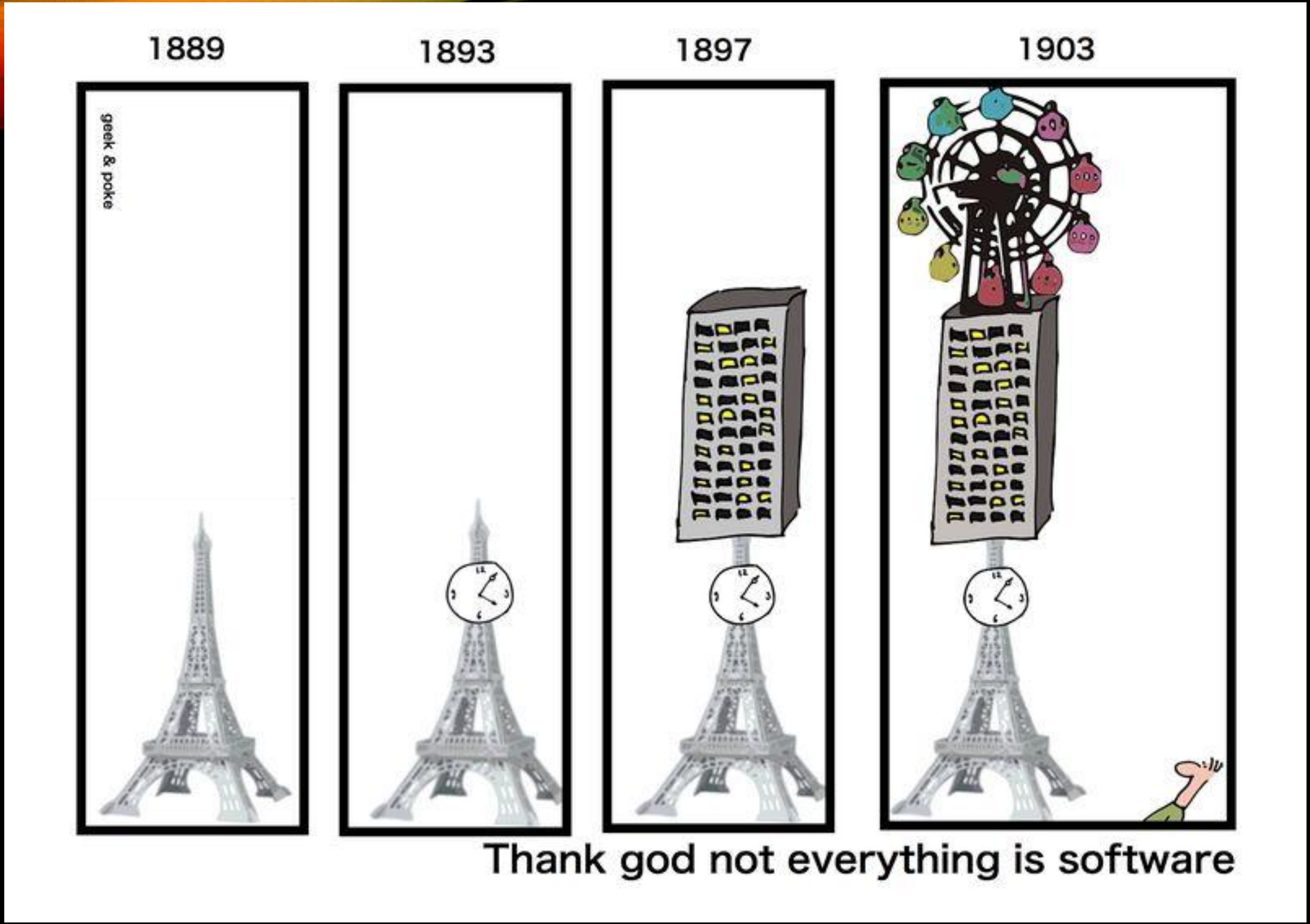
## Coding for Humans

The computer doesn't care what code looks like.
All that matters is that the code compiles.

But what about the humans?

## Why Do We Care?

There's no such thing as write-once code



http://geekandpoke.typepad.com/geekandpoke/2012/03/thank-god-not-everything-is-software.html

## Why Do We Care?

There's no such thing as write-once code

- Bug Fixes
- Business Changes
- Enhancements
- New Functionality

## Qualities

- Readable
- Maintainable
- Testable
- Elegant

## Blockers

- Ignorance
- Stubbornness
- Short-Timer Syndrome
- Arrogance
- Job Security
- Scheduling

## Blockers

Number one reason:

“I’ll clean it up later.”

Pro Tip: “Later” never comes.

## The Truth about Human Code

- Human readable code saves time.
- We can’t take a short-term view of software.
- We need to look at the lifespan of the application.

## Inspiration

- Rule of Thumb:

Imagine that the developer
who comes after you
is a homicidal maniac
who knows where you live.

-Unknown

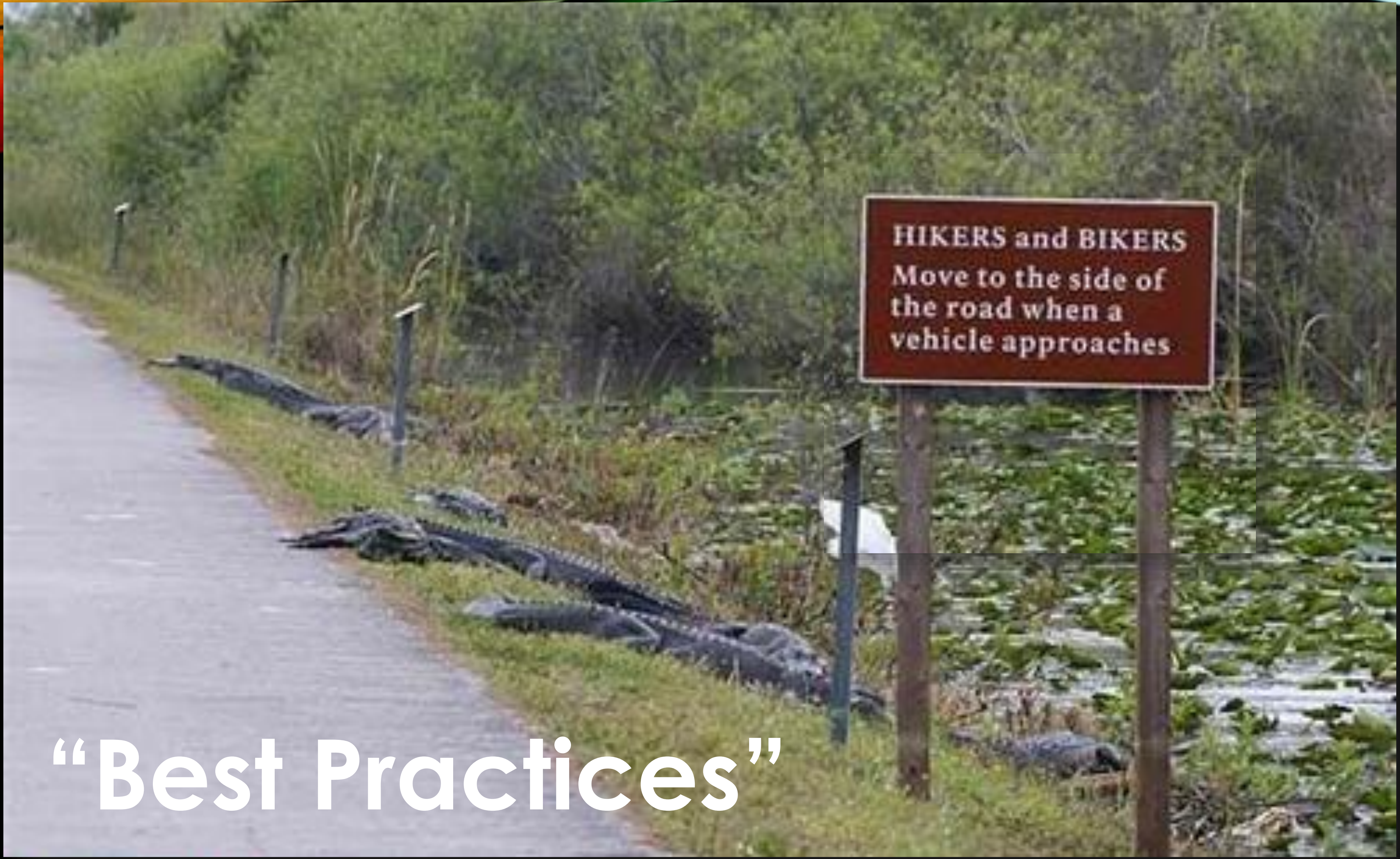
## The Next Developer

## This Might Take Awhile

## The Problem

- Readable (by mere mortals)
- Maintainable
- Testable
- Elegant

All of these qualities are subjective.



## “Best Practices”

@jeremybytes

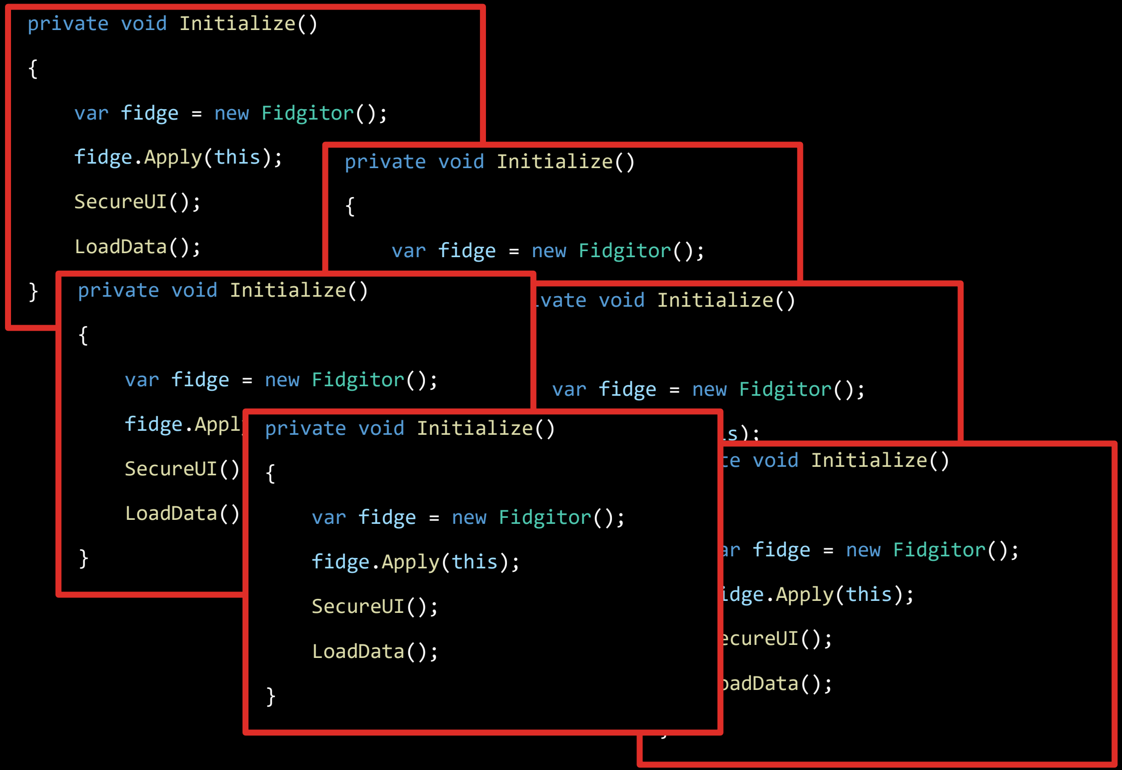
## The DRY Principle

# Don’t Repeat Yourself

- copy/pasta = spaghetti code

## Fixing a Memory Leak?

```
private void Initialize()
{
    var fidge = new Fidgitor();
    fidge.Apply(this);
    SecureUI();
    LoadData();
    free fidge;
}
```

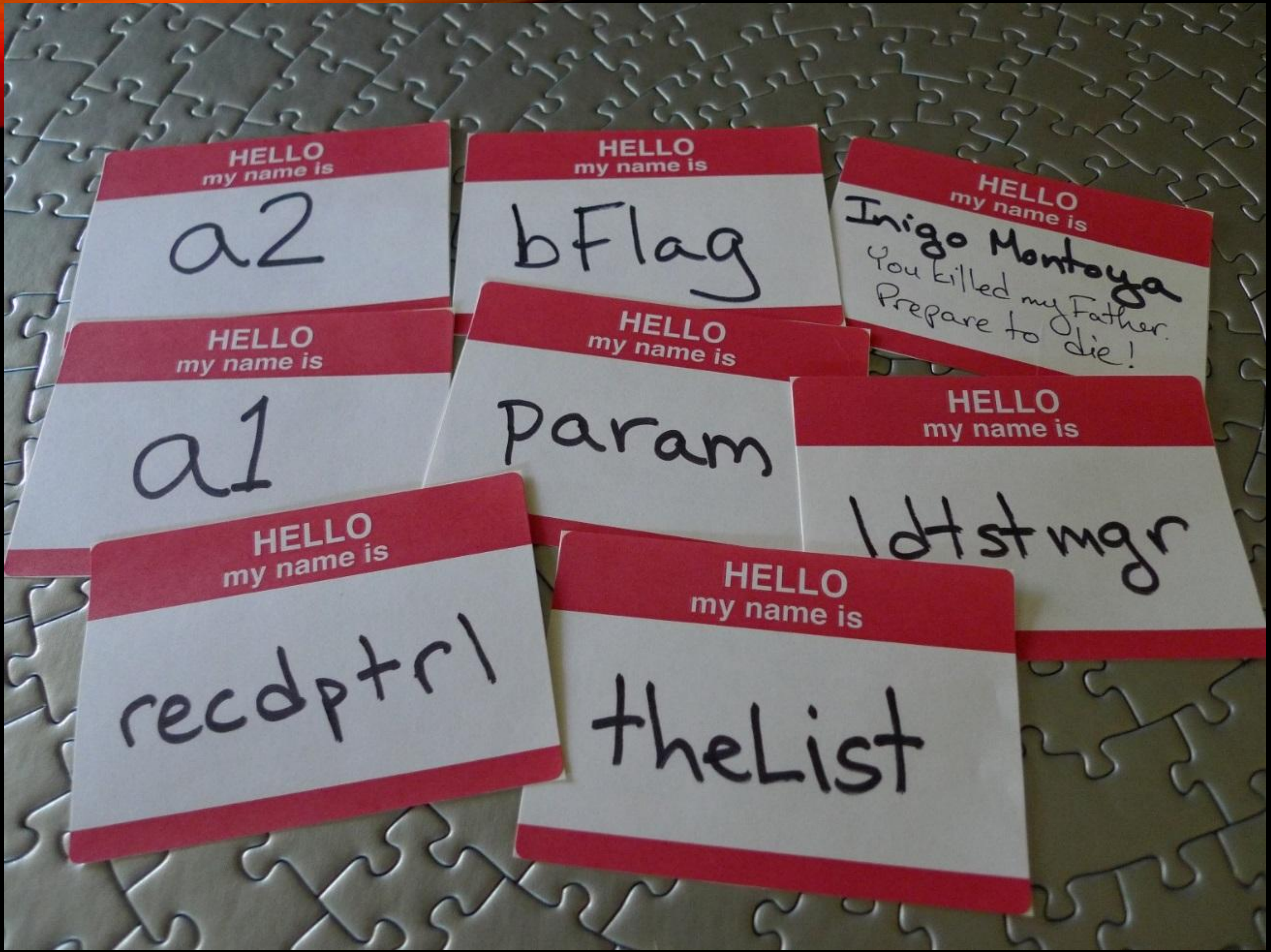


```
private void Initialize()
{
    var fidge = new Fidgitor();
    fidge.Apply(this);
    SecureUI();
    LoadData();
}
```

## Advice

- The first time you need similar functionality, copy/paste.
- If you need it in a third spot, consider creating a shared method.

## Naming

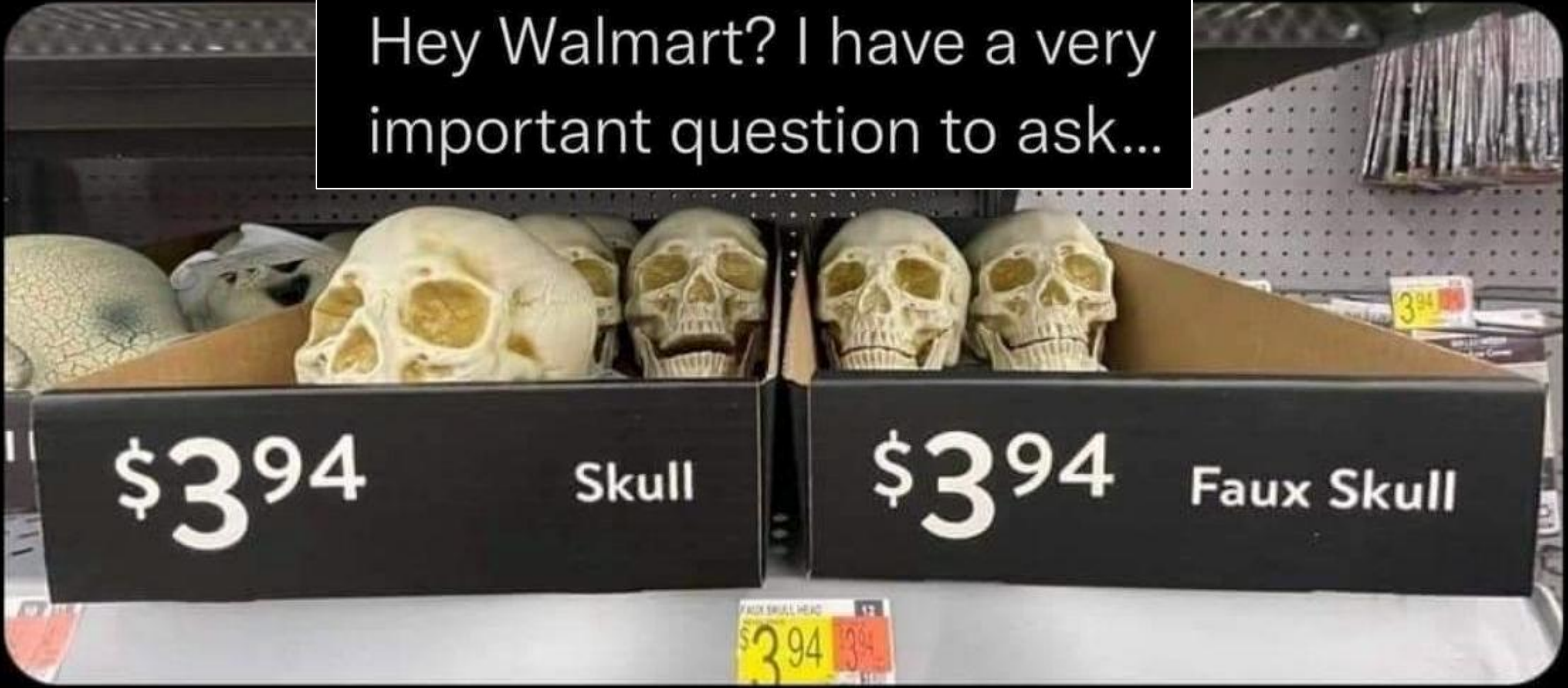

HELLO my name is
a2
HELLO my name is
bFlag
HELLO my name is
Inigo Montoya
You killed my Father.
Prepare to die!
HELLO my name is
a1
HELLO my name is
param
HELLO my name is
ldtstmgr
HELLO my name is
recdptrl
HELLO my name is
theList

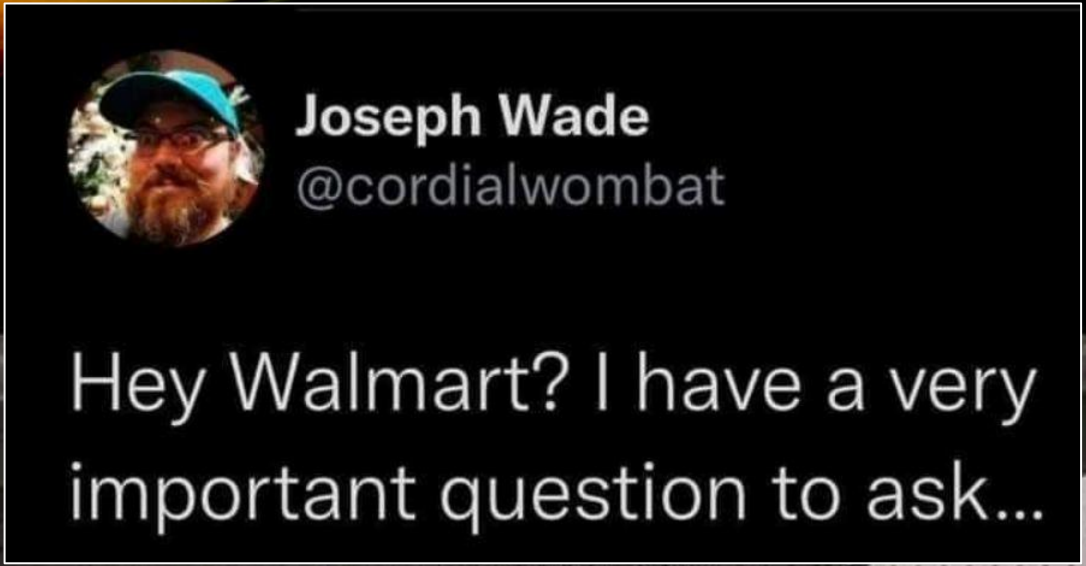
## Intentional Naming

- `theList`
  - Not very good
- `ProductList`
  - A bit better
- `ProductCatalog`
  - Good

## Naming

- Use Nouns for Variables, Properties, Parameters
  - `indexer`, `currentUser`, `PriceFilter`
- Use Verbs for Methods and Functions
  - `SaveOrder()`, `getDiscounts()`, `RunPayroll()`
- Pronounceable and Unambiguous
  - `recdptrl` = received patrol? record department role?




Joseph Wade
@cordialwombat

Hey Walmart? I have a very important question to ask...

## Naming Standard

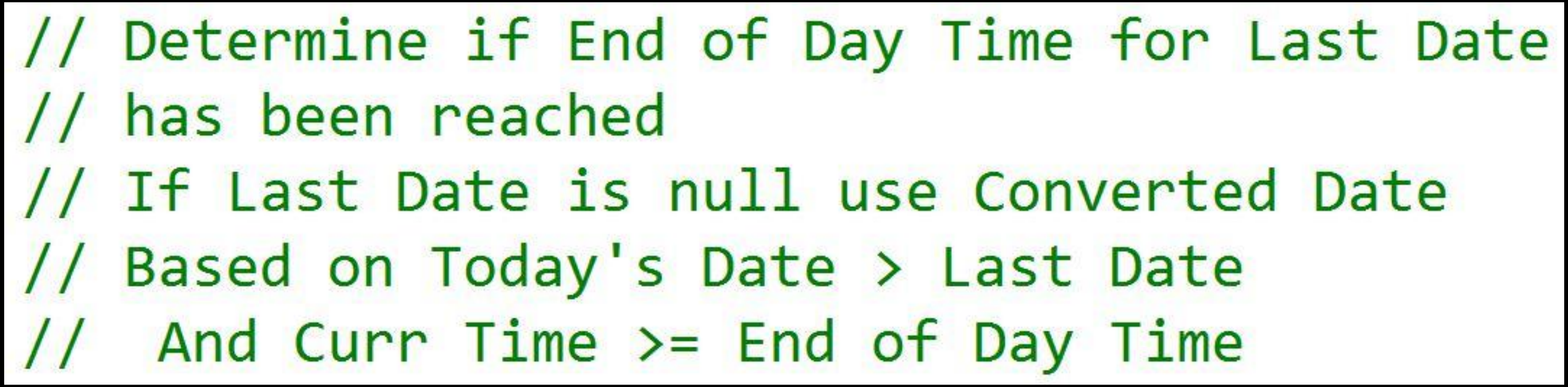
- camelCase?
- PascalCase?
- snake_case?
- kebab-case?

It doesn't matter

Have a Standard

Be Consistent

## Comments

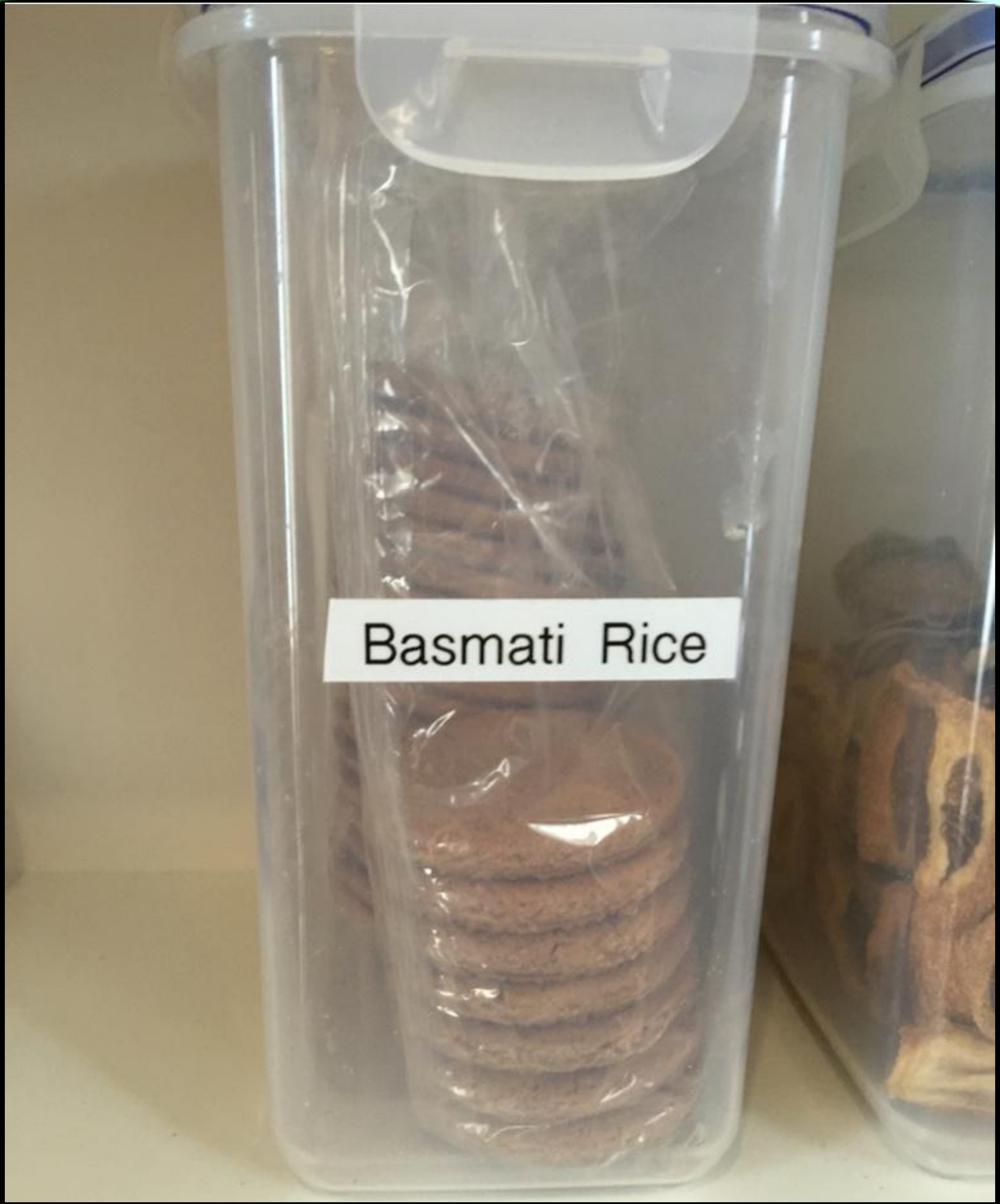


```
// Determine if End of Day Time for Last Date
// has been reached
// If Last Date is null use Converted Date
// Based on Today's Date > Last Date
//  And Curr Time >= End of Day Time
```

## Comments

- Rule #1: Comments lie
    - Code is updated or moved, but not the comments

## Comments Lie



## Comments

- Rule #1: Comments lie
  - Code is updated or moved, but not the comments
- Rule #2: Comments do not make up for bad code
  - If the code is that unclear, rewrite the code

## Good Comments

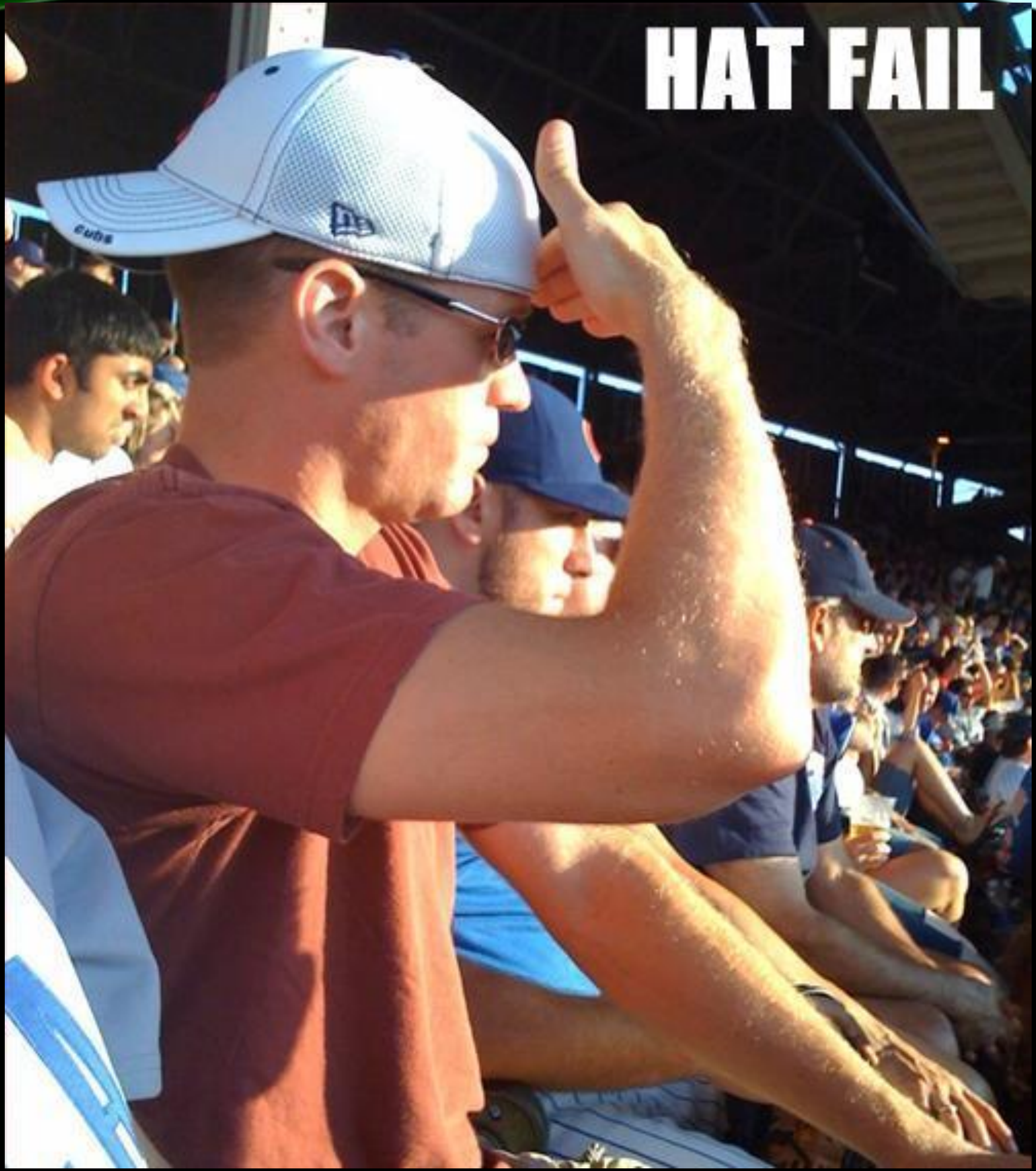
- Can be used to describe intent or clarification
  - Ex: // Sample input: Oct 5, 2015 - 13:54:15 PDT

- Can be used to give warnings or consequences
  - Ex:
    ```
    // We do a deep copy of this collection to make
    // sure that updates to one copy do not affect
    // the other
    ```

## Good Comments

- Can be used for TODOs
    - Especially useful when the IDE supports it
    - These should be temporary

## Know Your Tools



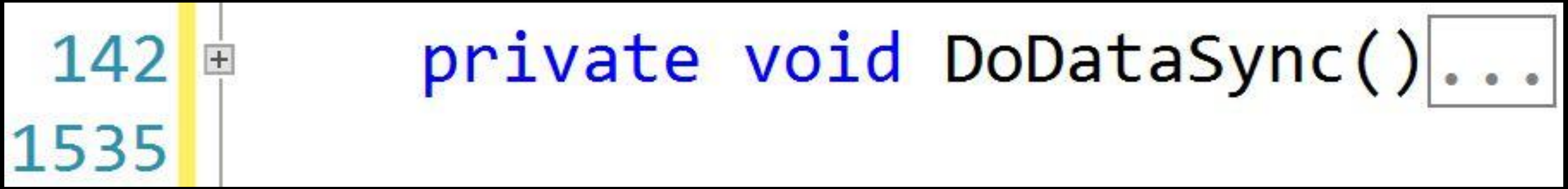
## Bad Comments

- Do not comment out code
  - Code no longer in use should be deleted
  - If needed, you can always retrieve it from source control

## Know Your Tools

@jeremybytes

## Functions and Methods



```
 142 +        private void DoDataSync()...
1535
```

@jeremybytes

## Functions and Methods

- Keep methods short
  - Should fit on a single screen
  - Prefer methods no longer than 10 lines

Do one thing!

## Multiple Levels of Methods

- High level
  - Overview of functionality
- Mid-level
  - More details, but not too deep
- Detail
  - The “weeds” of the functionality

## Work in Small Chunks

If you aren't writing incremental code, you are writing excremental code.

## What is Refactoring?

Making code better
without changing the functionality

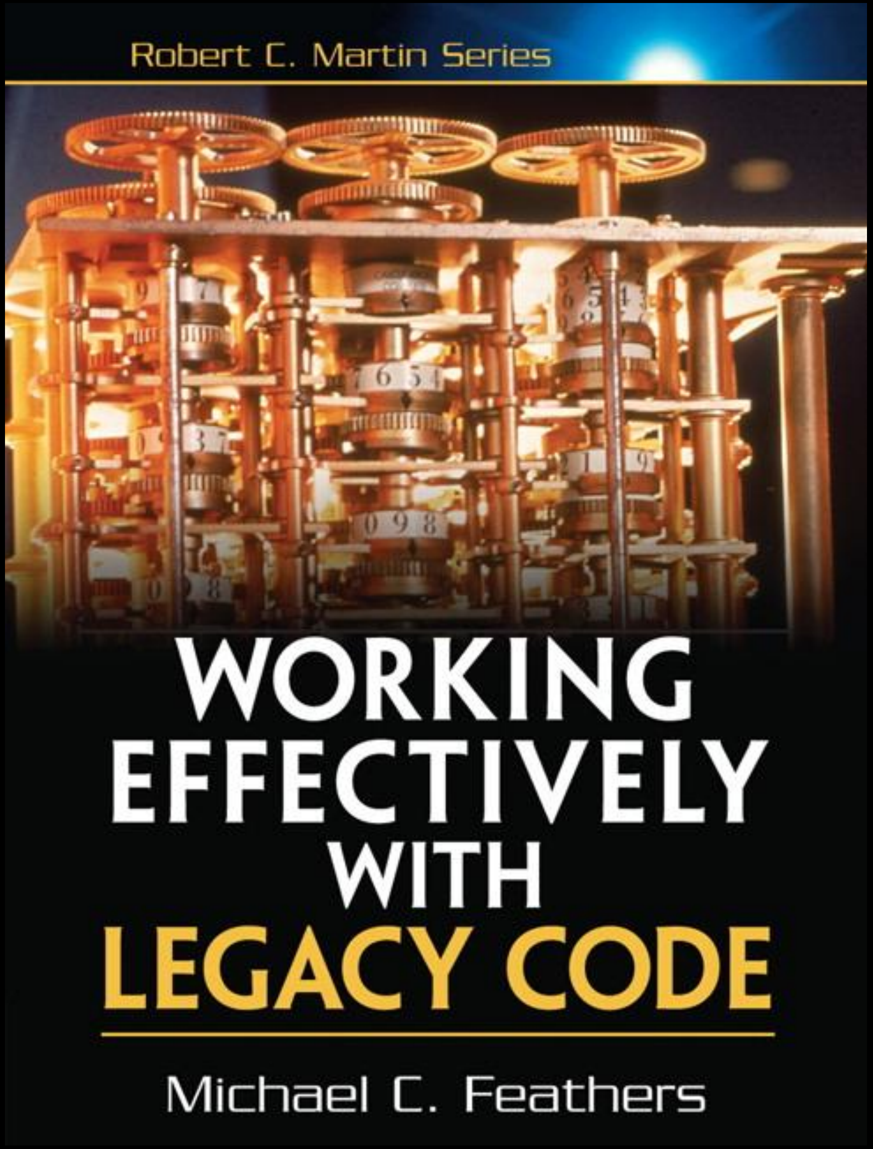
## Refactoring and Unit Testing

- If you don’t have unit tests,
  you don’t know what your code does.

- Refactoring Step 1:
  - Bring your code under test.
- Refactoring Step 2:
  - Safely and confidently update the code.

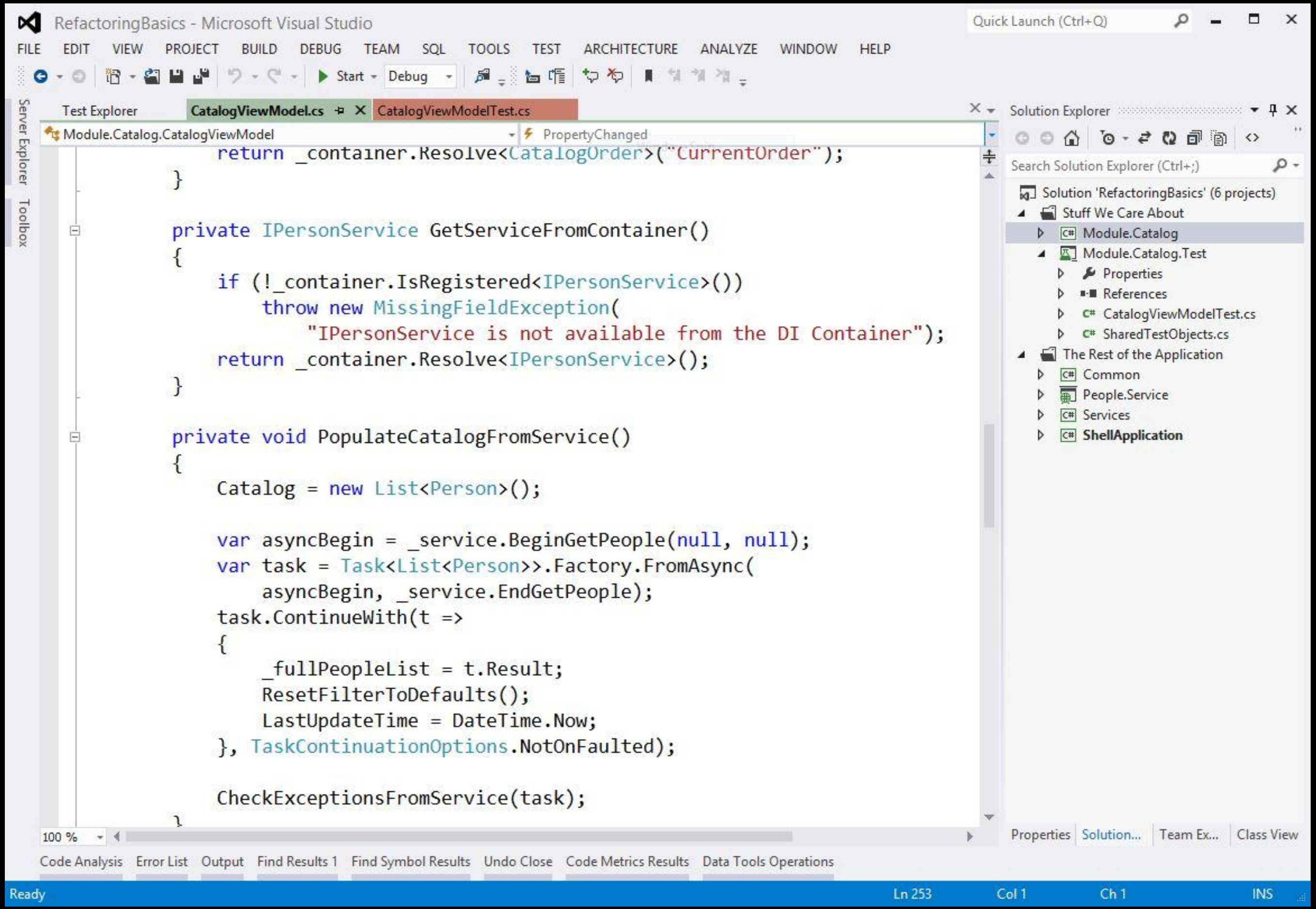
## Working Effectively with Legacy Code

- Michael C. Feathers



The Watcher

# Coding for Humans



```
        return _container.Resolve<CatalogOrder>("CurrentOrder");
    }

    private IPersonService GetServiceFromContainer()
    {
        if (!_container.IsRegistered<IPersonService>())
            throw new MissingFieldException(
                "IPersonService is not available from the DI Container");
        return _container.Resolve<IPersonService>();
    }

    private void PopulateCatalogFromService()
    {
        Catalog = new List<Person>();

        var asyncBegin = _service.BeginGetPeople(null, null);
        var task = Task<List<Person>>.Factory.FromAsync(
            asyncBegin, _service.EndGetPeople);
        task.ContinueWith(t =>
        {
            _fullPeopleList = t.Result;
            ResetFilterToDefaults();
            LastUpdateTime = DateTime.Now;
        }, TaskContinuationOptions.NotOnFaulted);

        CheckExceptionsFromService(task);
```

@jeremybytes

## Wrap Up

- There’s no such thing as write once code
- Be nice to the humans who have to change your code (it may be you)

## Thank You!

Jeremy Clark

- jeremybytes.com
- jeremy@jeremybytes.com
- github.com/jeremybytes

https://github.com/jeremybytes/code-is-for-humans