



Clean Code

Homicidal Maniacs Read Code, Too

Jeremy Clark
www.jeremybytes.com
[@jeremybytes](https://twitter.com/jeremybytes)

What is Clean Code?

- Readable
- Maintainable
- Testable
- Elegant



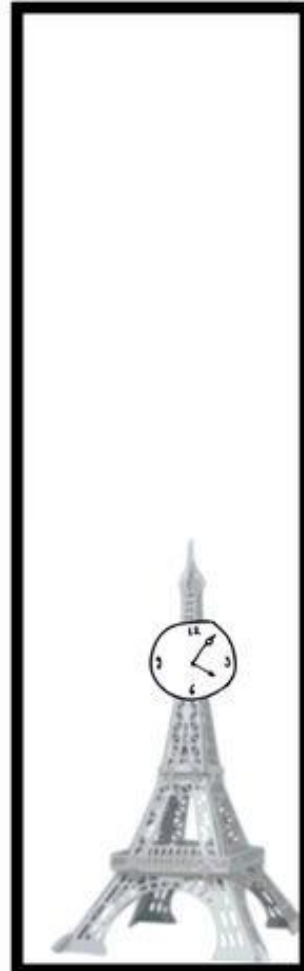
Why Do We Care?

There's no such thing
as write-once code

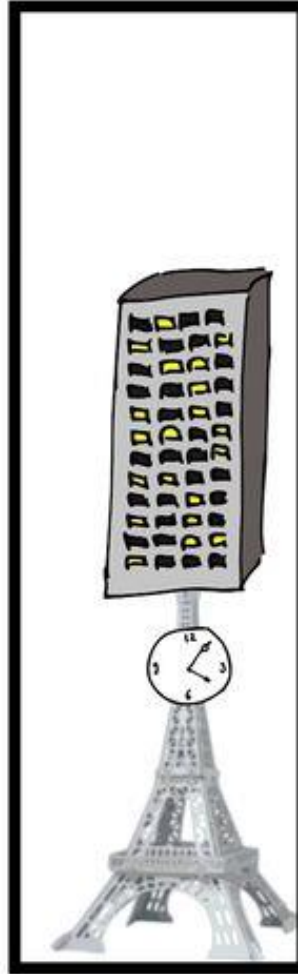
1889



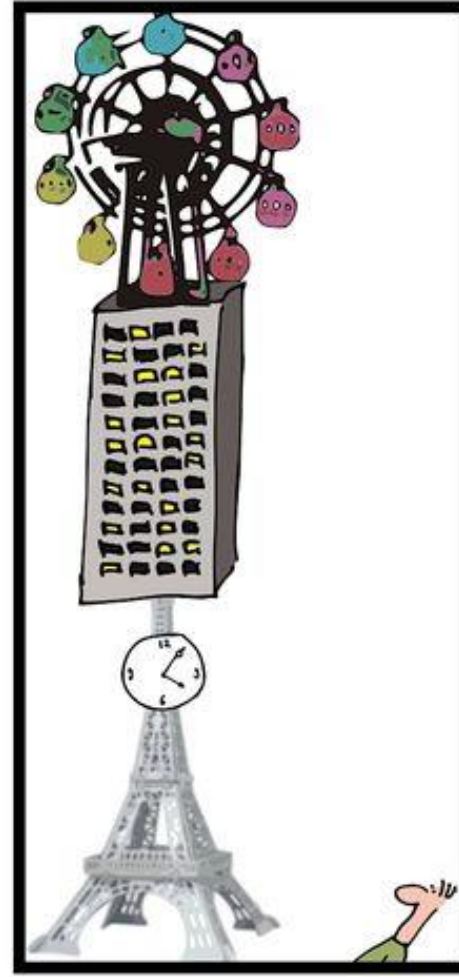
1893



1897



1903



Thank god not everything is software

Why Do We Care?

There's no such thing
as write-once code

- Bug Fixes
- Business Changes
- Enhancements
- New Functionality

What Prevents Clean Code?

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

What Prevents Clean Code?

Number one reason:

“I’ll clean it up later.”

Pro Tip: “Later” never comes.

The Truth about Clean Code

- Clean Code saves time.
- We can't take a short-term view of software.
- We need to look at the lifespan of the application.

How Do You Write Clean Code?

- Rule of Thumb:

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

-Unknown

The Next Developer



Jason

This Might
Take Awhile

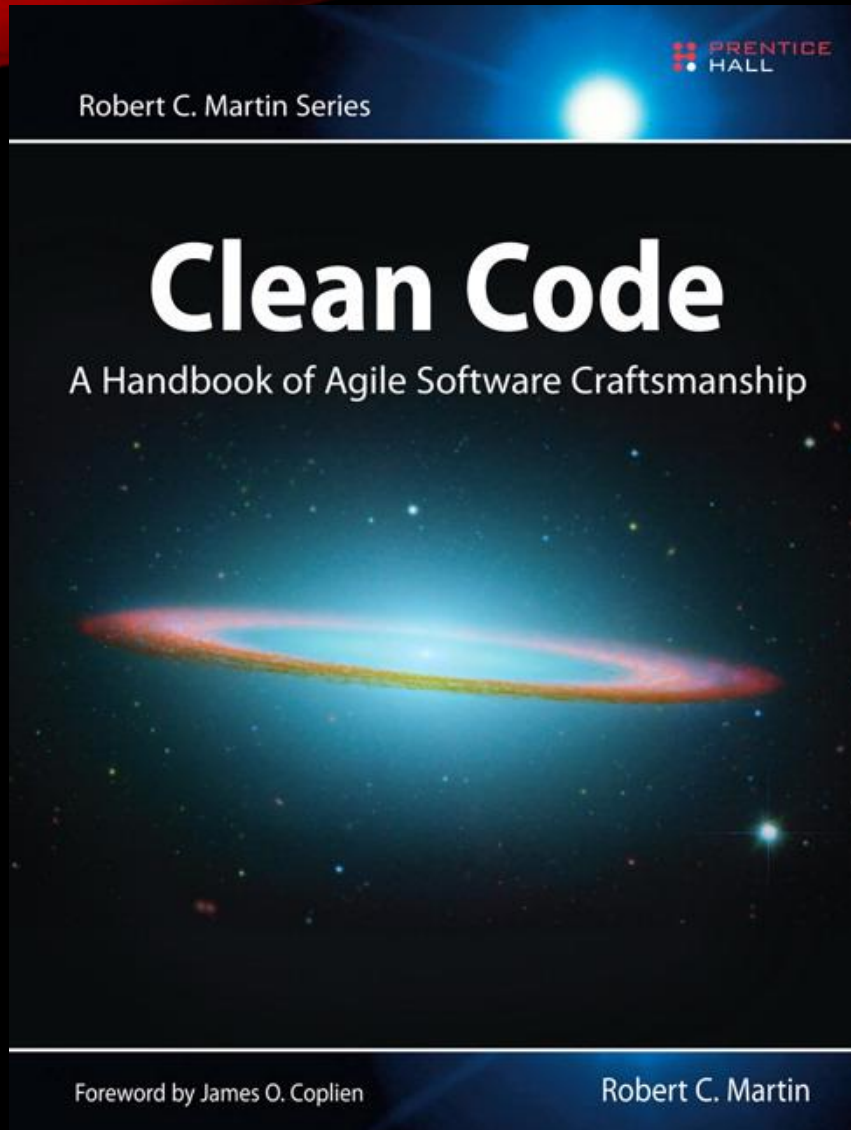


The Problem

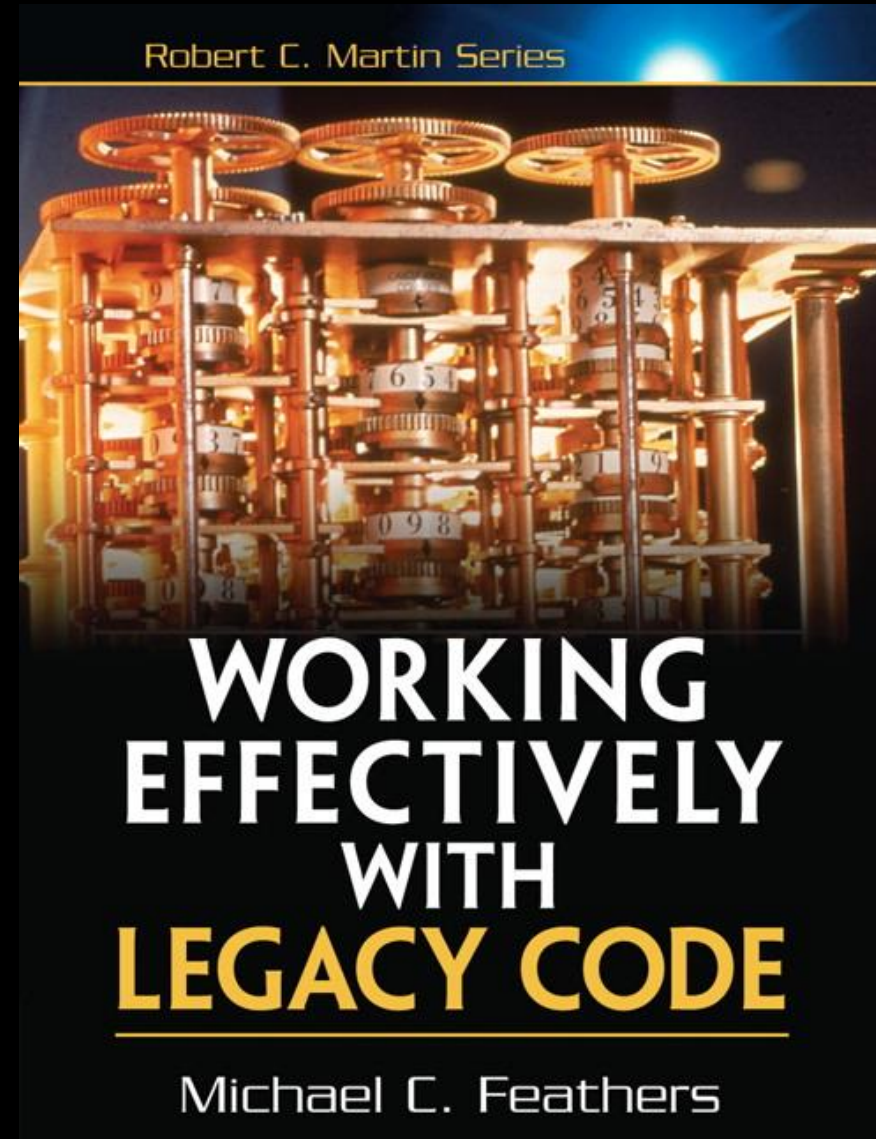
- Readable (by mere mortals)
- Maintainable
- Testable
- Elegant

All of these qualities are subjective.

Robert C. Martin



Michael C. Feathers





The Dry Principle

Don't Repeat Yourself

- copy/pasta = spaghetti code

Naming



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
 - `recdptr1` = received patrol? record department role?

Naming Standard

- Camel Case?
- Pascal Case?
- Lower Case with Underscores?

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`

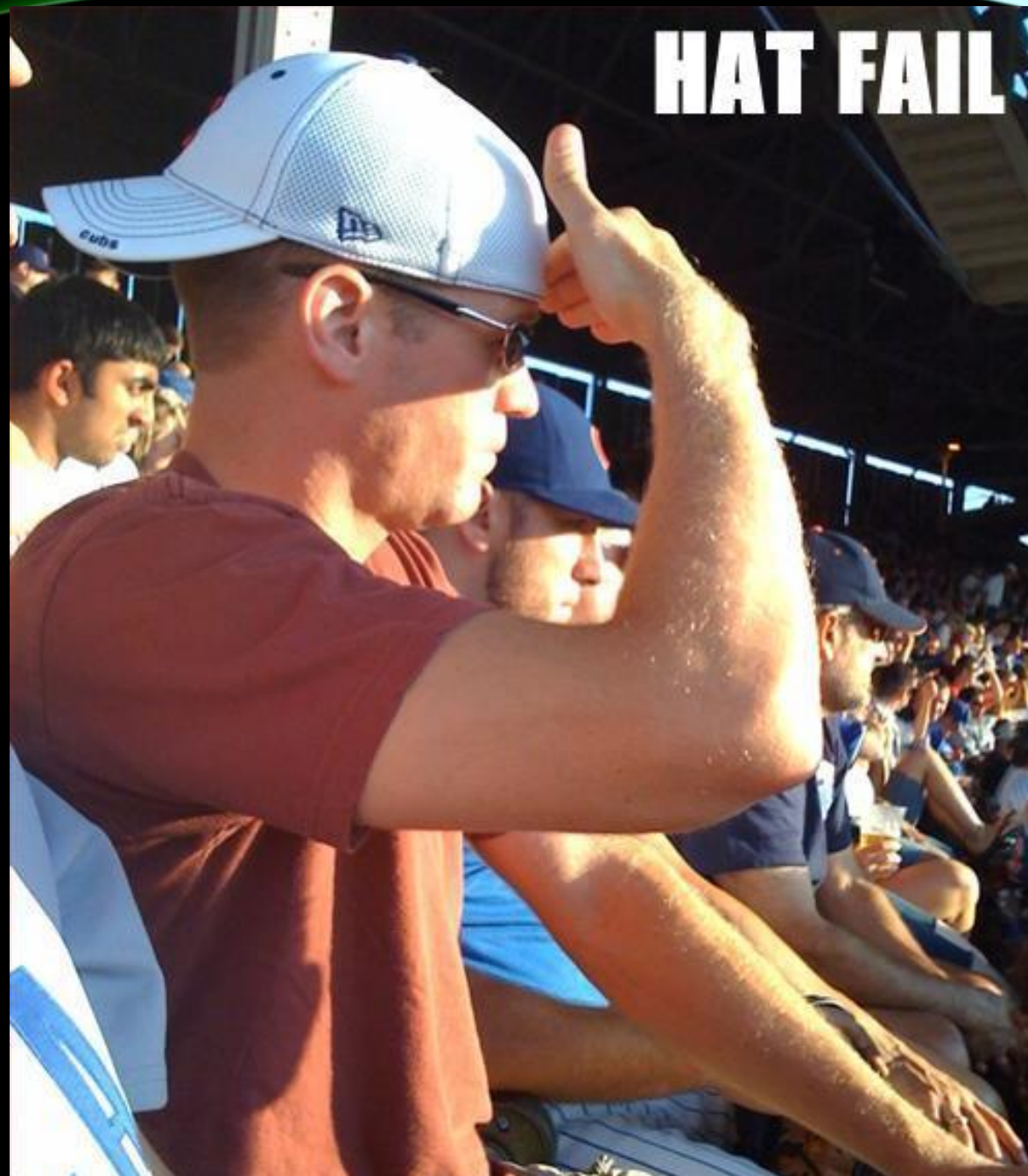
Good Comments

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

Bad Comments

- Avoid “journaling” comments
 - Ex: `// 03/20/1996 - jjc - Added tax calculation`
 - This is what source control is for: Who, What, When

Know Your Tools



Bad Comments

- Avoid “journaling” comments
 - Ex: `// 03/20/1996 - jjc - Added tax calculation`
 - This is what source control is for: Who, What, When
- Avoid “noise” comments
 - Ex: `// Default constructor`

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



Functions and Methods

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

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

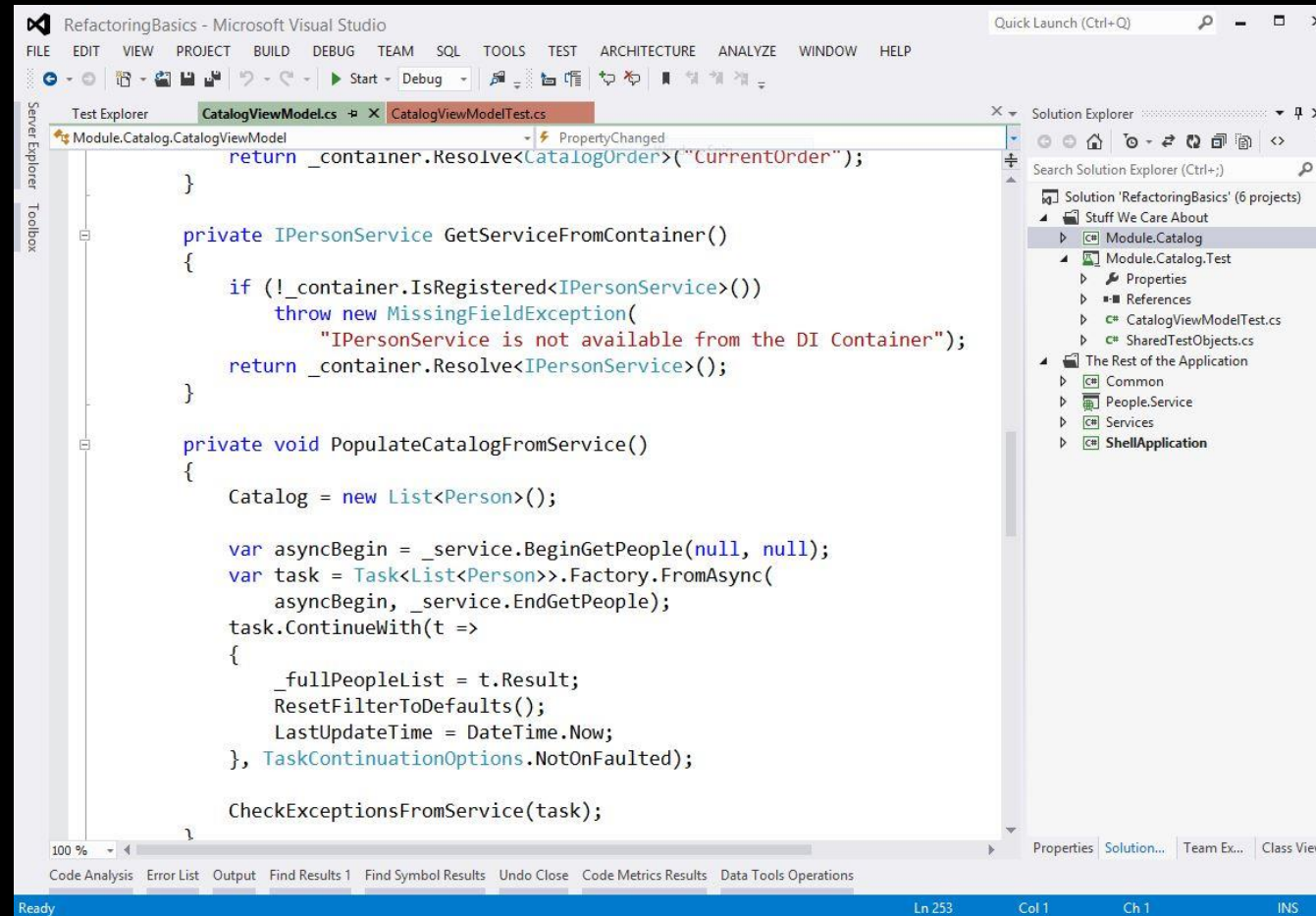
- If you don't have unit tests, you don't know what your code does.
- Refactoring Step 1:
 - Bring your code under test.
- Refactoring Step 2:
 - Safely and confidently update the code.

The Watcher



Jason

Making Code Cleaner



```
RefactoringBasics - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Start - Debug
Test Explorer CatalogViewModelTest.cs CatalogViewModelTest.cs
Module.Catalog.CatalogViewModel
Property Changed
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);
}
100 %
Code Analysis Error List Output Find Results 1 Find Symbol Results Undo Close Code Metrics Results Data Tools Operations
Ready Ln 253 Col 1 Ch 1 INS
```

Be a Clean Code Advocate

The Boy Scout Rule

Always leave the campground
cleaner than you found it.

The Clean Coder Rule

Always leave the code
cleaner than you found it.



Thank You!

Jeremy Clark

- <http://www.jeremybytes.com>
- jeremy@jeremybytes.com
- @jeremybytes