Refactoring Reminders

Don’t get lost in the trees

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

Why

Readability

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

Ideas

Communication

~~(Business)~~ Logic

Maintainability

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

When

Rule of Three

“Two instances of the same code do not require refactoring - three, yes.”

1. When you’re doing something for the first time, just get it done.
2. When you’re doing something similar for the second time, cringe at having to repeat but do the same thing anyway.
3. When you’re doing something for the third time, start refactoring.

Scout’s Rule

“Always leave the campsite cleaner than when you found it.”

WRF

Work > Right > Fast

Make It Work, then

Make It Right, then

Make It Fast

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

How

Code Smells (?)

Code Smells - Bloaters

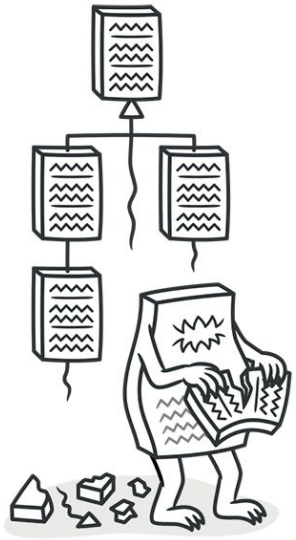
Photo from refactoring.guru

Bloaters

Bloaters are code, methods and classes that have increased to such gargantuan proportions that they are hard to work with. Usually these smells do not crop up right away, rather they accumulate over time as the program evolves (and especially when nobody makes an effort to eradicate them).

- § **Long Method**
- § **Large Class**
- § **Primitive Obsession**
- § **Long Parameter List**
- § **Data Clumps**

Code Smells - Abusers



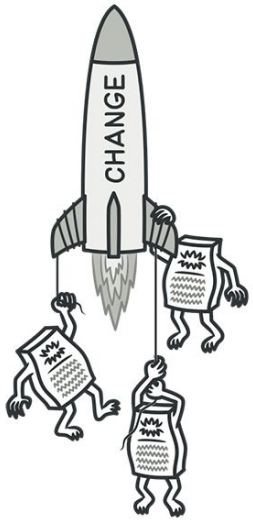
Object-Orientation Abusers

All these smells are incomplete or incorrect application of object-oriented programming principles.

- § Alternative Classes with Different Interfaces
- § Refused Bequest
- § Switch Statements
- § Temporary Field

Photo from refactoring.guru

Code Smells - Preventers



Change Preventers

These smells mean that if you need to change something in one place in your code, you have to make many changes in other places too. Program development becomes much more complicated and expensive as a result.

- § Divergent Change
- § Parallel Inheritance Hierarchies
- § Shotgun Surgery

Photo from refactoring.guru

Code Smells - Dispensables

Dispensables

A dispensable is something pointless and unneeded whose absence would make the code cleaner, more efficient and easier to understand.

- § Comments
- § Duplicate Code
- § Data Class
- § Dead Code
- § Lazy Class
- § Speculative Generality

Photo from refactoring.guru

Code Smells - Couplers

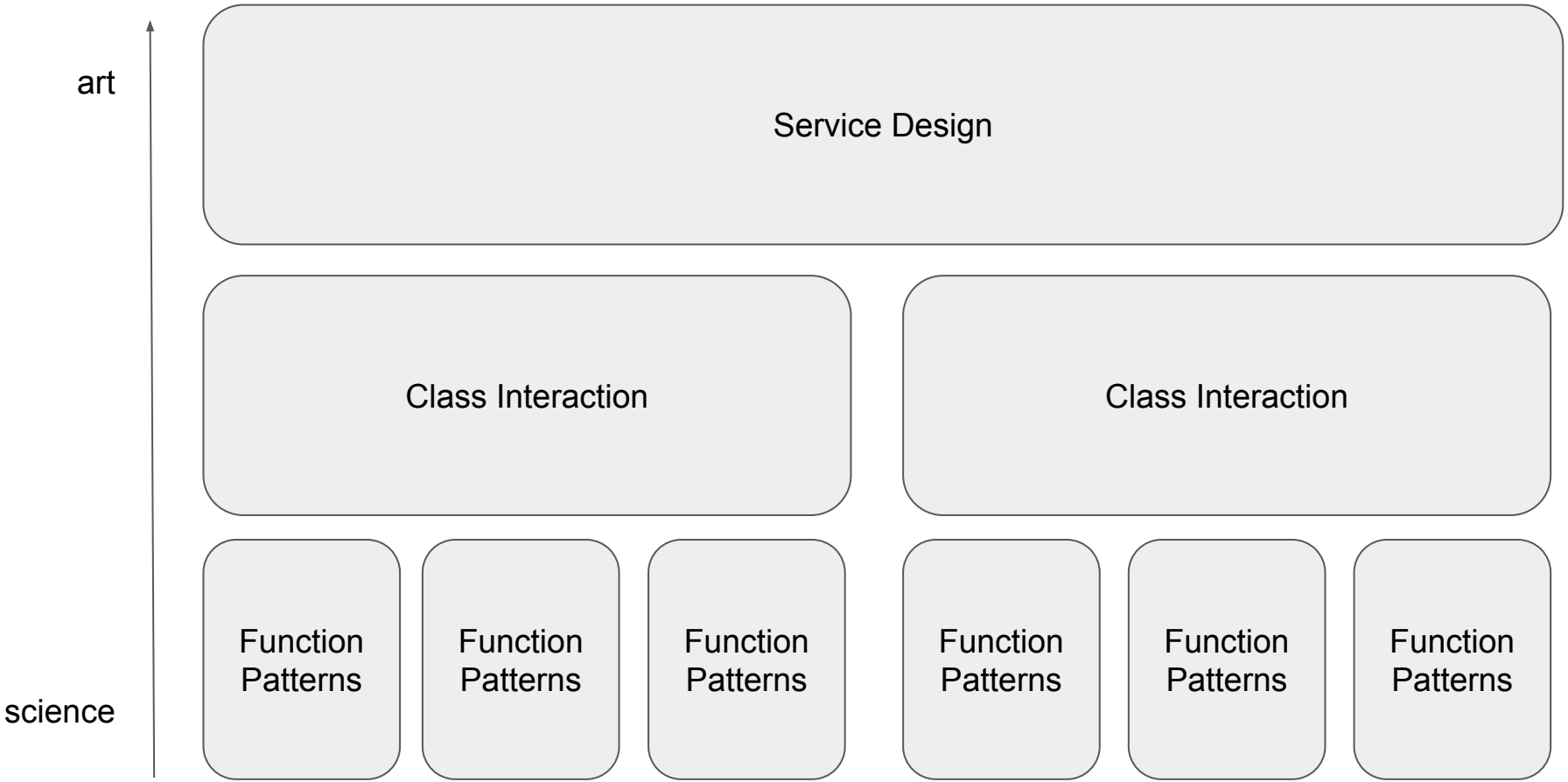
Couplers

All the smells in this group contribute to excessive coupling between classes or show what happens if coupling is replaced by excessive delegation.

- § Feature Envy
- § Inappropriate Intimacy
- § Incomplete Library Class
- § Message Chains
- § Middle Man

Photo from refactoring.guru

But how many of these sound like judgement calls?


art
Service Design
Class Interaction
Class Interaction
Function Patterns
Function Patterns
Function Patterns
Function Patterns
Function Patterns
Function Patterns
science

Use Only What is Needed, Not More

Less is More!

Principle of Least Astonishment

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

Be Idiomatic

Make use of the tools you have

Fail Loudly with trace

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

Would I be able to understand this in one week, two months, three years?

Delete
Rename
Move

Oh! And prefer Renaming a file than Deleting and Adding.

For it to appear as a rename in Git - rename the file in one commit, then modify in another if necessary.

Start Small

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

Communicate!

Before, please

Guiding Principles

- No change in functionality
- Tests continues passing and build is green
- Can do/stop anytime (!)

Principles > Practices

One More Thing

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

On nulls

Let’s try to avoid them!

Samantha Wong - Dashboard Learning Sessions - 20 May 2021

As an Input:

- `null` should not be allowed as a valid input
  - There should be an explicit empty/base case for that.
- Making variables non-nullable makes application behaviour predictable
- `null` should only be allowed if we cannot help something to be null if things goes wrong

On nulls

[cont]

As an Output:

- It’s the easy go-to for an empty object

Links

References

https://www.industriallogic.com/img/blog/2005/09/smellstorefactorings.pdf

https://flylib.com/books/en/1.476.1/

https://refactoring.guru/

CI/CD

- Small, frequent changes -> lowering deployment risk
- Build pipelines need to be well maintained -
  - Easy to find where are the mistakes

End