Mentimeter

Instructions

Mentimeter

How is it going?

0 responses

Mentimeter

Do you have problems with practice?

Agenda

- Recall last session
- Discuss current topic
- Practice

Total Recall

Mentimeter

What do you remember most from the previous session?

Mentimeter

What is the purpose of the YAGNI principle?

0 ✓
To prevent over-engineering

0 ✗
To improve software performance

0 ✓
To reduce code complexity

0 ✗
To deliver software faster

Mentimeter

YAGNI helping-questions

- Is it 100% sure that the implementation is needed?
- What are the investment costs?
- How long does the implementation take?
- What are the opportunity costs?

Mentimeter

Advice

DEFINE requirements!!!

Start with abstractions

Design iteratively and incrementally

Add changes or perform refactoring according to the requirements

Mentimeter

KISS goals and benefits

0 responses

Mentimeter

KISS goals

- Simplicity
- Clarity
- Efficiency
- Usability
- Flexibility
- Risk Reduction

Advice

Keeping everything as simple as possible is in 90% case the most important dev strategy

Mentimeter

DRY is about

0 not duplicating code

0 not duplicating logic

0 not duplicating constants

Discuss current topic

Mentimeter

Random fact

Two thirds of the inhabitants of our planet have never seen real snow in their lives.

Mentimeter

Code smells

Mentimeter

Have you looked through code smells materials?

0

yes

0

no

Mentimeter

Code smells is about

0

early bugs

0

smell of burning processor while your app is running

0

symptoms of a bad code

Mentimeter

Code smells forms:

- Duplicated code: Repeated code blocks performing the same functionality
- Long and complex functions: Functions that are difficult to understand and maintain
- Large classes : Classes that do too much and become overly complex
- Magic numbers: Using hardcoded values without explanation, making code less adaptable

Mentimeter

What are the symptoms of code smells in method level?

Methid level

- Naming convention (descriptive namings)
- Too big function, too many actions within it
- Excessive parameters (DTOs vs primitives)
- Excessive return data (DTO vs tupple or primitive)
- One line code, too much of functional programming (split logically to make it clear)

Mentimeter

What are the symptoms of code smells in class level?

Class level

- Class GOD
- Feature envy (excessively uses methods from not dependency class)
- Inappropriate intimacy (dependency on implementation or private members of another class)
- Literal usage (hardcoded values: difficult to maintain and modify)
- Data clumping (primitives not objects as fields)

Mentimeter

What are the symptoms of code smells in application level?

App level

- Shotgun surgery (duplicated code blocks)
- Contrived complexity (do not use design patterns just because they exist)

Mentimeter

Design smells.

what?!

Mentimeter

How to recognize software design "smell"? (improper abstractions, encapsulation mistakes, incorrect modularization, hierarchy mismanagement)

Mentimeter

Forms of smelling design

- Improper abstractions
- encapsulations misunderstanding
- incorrect modularization
- hierarchy mismanagement

Bad abstractions

- Details not abstract
- Multifaceted abstractions (SRP violation)
- Duplicated abstractions (same or close responsibilities)
- Incomplete abstraction (not fully support responsibility)

Encapsulations misunderstanding

- Deficient encapsulation (poorly encapsulated state)
- Unrestrained encapsulation (abstraction’s state is globally visible)
- Unexploited encapsulation (use endless if-else instead of exploiting existing type variation -> solve with polymorphism)

Incorrect modularization

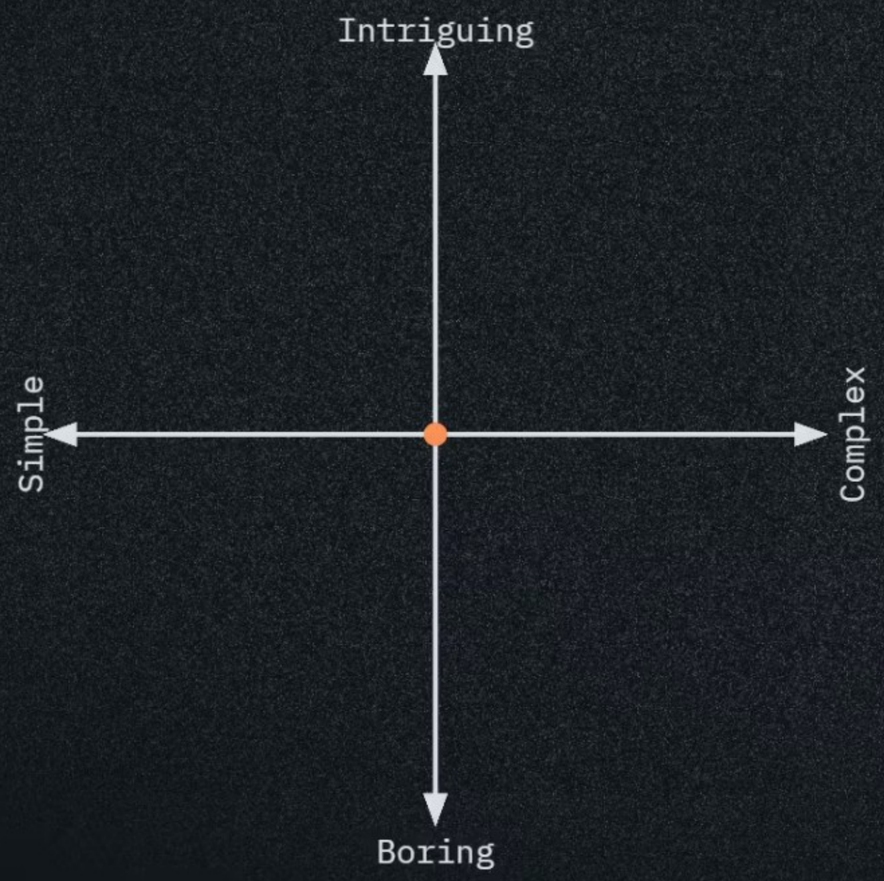
- Insufficient decomposition (require further decomposition)
- Broken modulariation (closed methods and not withing one class)
- Cyclically dependent abstractions (too tight coupled)

Mentimeter

Hierarchy management

- Repeatedly inherited abstractions
- Broken hierarchy (no AS-A relation)
- Too complex
- Cyclic hierarchy (super depends on subtype)

Express your opinion



1 What do you think about current course

Mentimeter

10 minute break

Refactoring

Mentimeter

What is the purpose of the refactoring?

0 Improve code

0 Enhance code

0 Fix bugs

Refactoring aimed

- Maintainable code
- Optimized code

Mentimeter

What to do before refactoring?

0 responses

Mentimeter

Check list before refactoring

- Functionality is well covered with tests
- You are not implementing a feature during refactoring
- Apply des-patterns only where necessary
- Research how to use design pattern rules and principles

Mentimeter

What antipatterns do you know??

What I remember

- Boat anchor - i will use it in future (opposite to yagni)
- Dead code (processed result never used)
- God object
- Golden hammer (one pattern\method\style for all cases) -> every problem has it's own solution
- Spaghetti code (no structure)

Few more

- MAGIC NUMBERS
- Yoda conditions (true==isProcessed or isProcessed==true)
- Excessive Interface Complexity (oppositte for ISP)
- Overuse of Singleton Pattern
- Overuse of Primitive Types
- Copy Paste Programming

Mentimeter

<wanna get deeper with antipatterns>?

https://sourcemaking.com/antipatterns/software-development-antipatterns

Mentimeter

Q&A part

0 questions
0 upvotes