Mentimeter

# Instructions

Mentimeter

# How is it going?

0 responses

Mentimeter

# Do you have problems with practice?

Mentimeter

# Agenda

- Recall last session
- Discuss current topic
- Practice

# Total Recall

Mentimeter

# What do you remember most from the previous session?

Mentimeter

# DIP - IoC - DI

DI is when you create objects separately and put one into another (i.e. inject): a pattern that let's you test, maintain code easier.

IoC helps you to do DI using instructions (e.g. xml), store objects, reuse, control lifecycle (main player is ioc-container).

DIP provides rules to setup modules\classes\objects and their relations (a principle with rules)

Mentimeter

# Disadvantages of the MVC Design Pattern

- Complexity: Implementing MVC can add complexity to code, especially for simple apps, leading to overhead in dev
- Learning Curve: Devs need to understand the concept of MVC and how to implement it effectively
- Overhead: communication between components can lead to overhead, affecting the performance of the app
- Potential for Over-Engineering: In some cases, devs may over-engineer the app by adding unnecessary abstractions and lay
- Increased File Count: MVC can result in a larger number of files and classes compared to simpler architectures

Mentimeter

# What is the observer pattern used for in MVC?

Mentimeter

# What is the facade pattern used for in MVC?

Mentimeter

# Discuss current topic

Mentimeter


# Random fact

About 500 meteorites of reasonable size hit the Earth's surface every year

Mentimeter

# Have you looked into yagni\kiss\dry?

0


yes

0

no

Mentimeter

# YAGNI

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

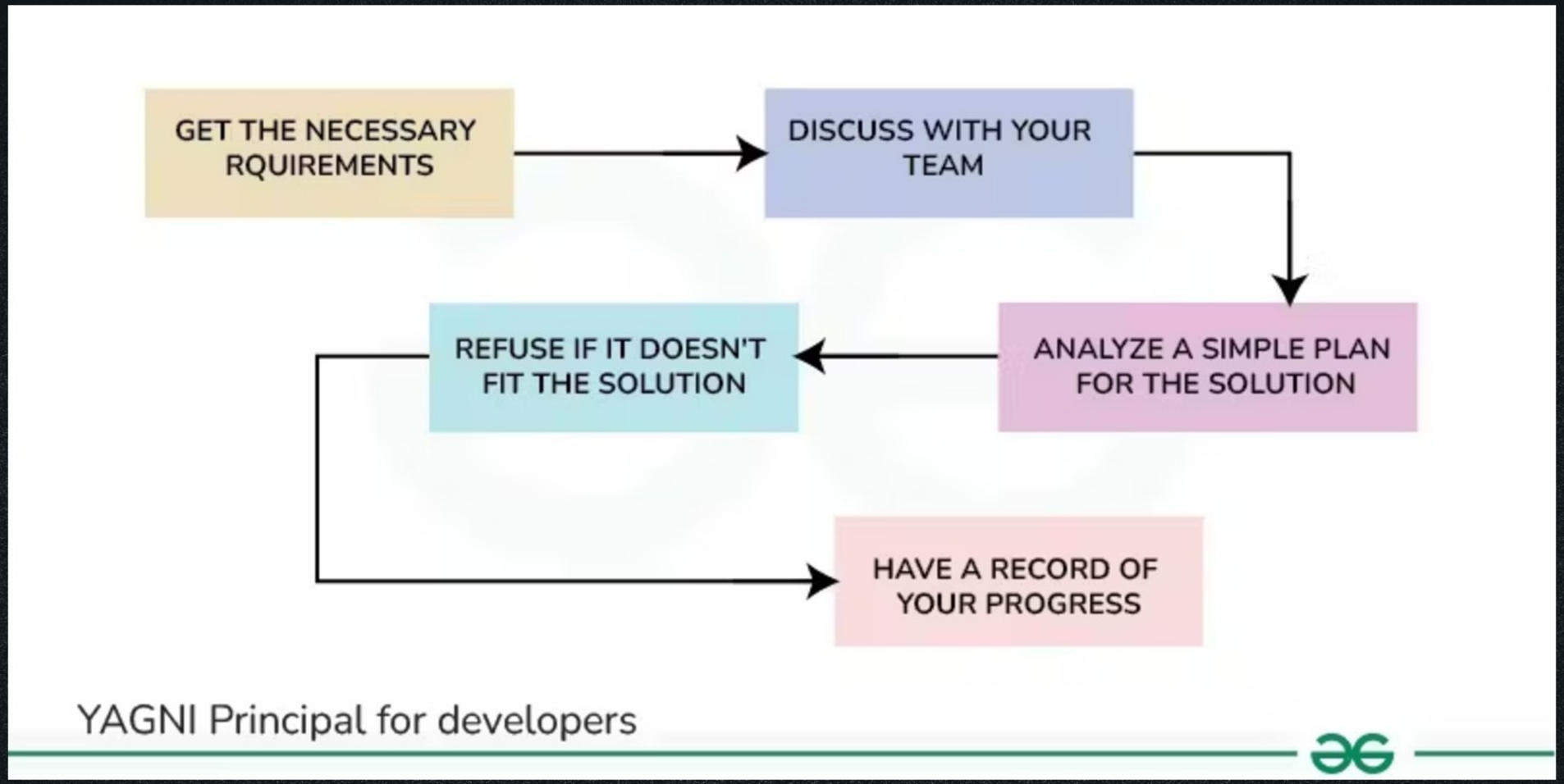
# What rules you may use to follow YAGNI?

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




# How do I use YAGNI in my code?

Mentimeter

# yagni examples

https://github.com/AndreiRohau/itpu-design-patterns-09-2024/blob/main/ws-06-yagni-kiss-dry-die/src/main/java/com/arohau/yagni/ex0/Main.java

Mentimeter

# YAGNI benefits

0 responses

Mentimeter

# yagni benefits

- Faster Development
- Simplicity
- Flexibility
- Reduced Risk
- User Focus
- Cost Savings
- Improved Maintainability
- Better User Experience

Mentimeter

# What are the criticisms of YAGNI?

Some people say that YAGNI has a downside. They argue that if you only think about what you need right now and ignore potential future needs, you might end up having to redo a lot of your work later when new requirements pop up.

Mentimeter

# What is the difference between SOLID and YAGNI?

SOLID expects you to have an idea, even if it’s just a bit, about how the code might change in the future, especially with Single Responsibility Principle (SRP). This is like being hopeful that you can predict some things. On the flip side, YAGNI assumes that most of the time, you don’t know where the code is headed in the future. It’s like being a bit doubtful about our ability to predict.

Mentimeter

# Advice

DEFINE requirements!!!

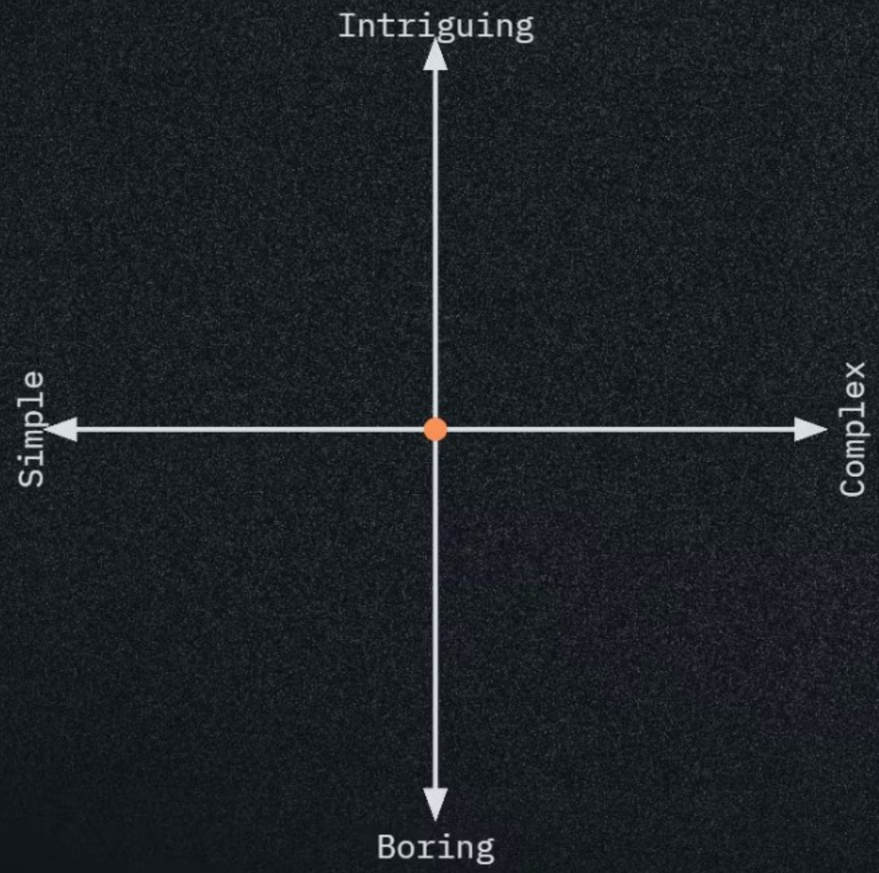
Start with abstractions

Design iteratively and incrementally

Add changes or perform refactoring according to the requirements

Mentimeter

# Express your opinion




1 What do you think about current course

Mentimeter

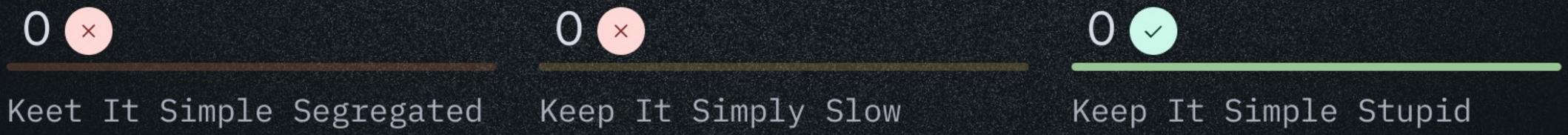
# 10 minute break

Mentimeter

# The KISS Principle

Mentimeter

# KISS acronym stands for



Mentimeter

# Other known 'synonyms'

- Keep It Short and Simple
- Keep It Simple and Straightforward
- Keep It Super Simple
- Keep It Silly Simple
- Keep It Simple and Stupidly Obvious
- Keep It Simple, Smart, or Stupid
- Keep It Stupid Simple and Streamlined

Mentimeter

# KISS goals and benefits

0 responses

Mentimeter

# KISS goals

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

Mentimeter

# KISS benefits

- Maintainability
- Debugging
- Scalability
- Performance
- Reduced Technical Debt

Mentimeter

# Name steps which help to Apply KISS Principle

Mentimeter

1. Identify Core Objectives
    - Clearly define the problem or objective you’re addressing.
    - Identify the essential goals and requirements.
2. Focus on Essentials
    - Prioritize essential features or components necessary to achieve objectives.
    - Avoid unnecessary embellishments or functionalities.
3. Simplify Design and Workflow
    - Streamline processes and workflows to minimize complexity.
    - Eliminate redundant steps or unnecessary complications.
4. Prioritize Clarity and Understandability
    - Ensure that solutions are clear and easy to understand for all stakeholders.
    - Use simple and straightforward language in documentation and communication.
5. Iterate and Refine
    - Continuously review and refine solutions to simplify further.
    - Seek feedback and iterate based on insights gained.

Mentimeter

1. Use Simple Tools and Techniques
    - Choose straightforward tools and methodologies that align with the KISS principle.
    - Avoid unnecessary complexity in tooling and technology choices.
2. Test for Simplicity
    - Evaluate solutions against the KISS principle to ensure simplicity.
    - Assess whether each component or feature is truly necessary.
3. Maintain Pragmatism
    - Balance simplicity with other considerations such as functionality and scalability.
    - Be pragmatic in decision-making and prioritize simplici
4. Educate and Advocate
    - Educate team members and stakeholders about the benefits of simplicity.
    - Advocate for the adoption of the KISS principle within your organization or team.
5. Lead by Example
    - Demonstrate the effectiveness of simple solutions through your actions and decisions.
    - Inspire others to embrace simplicity using expmle

Mentimeter

# KISS examples

https://github.com/AndreiRohau/itpu-design-patterns-09-2024/blob/main/ws-06-yagni-kiss-dry-die/src/main/java/com/arohau/kiss/ex0/kiss.md

Mentimeter

# Advice

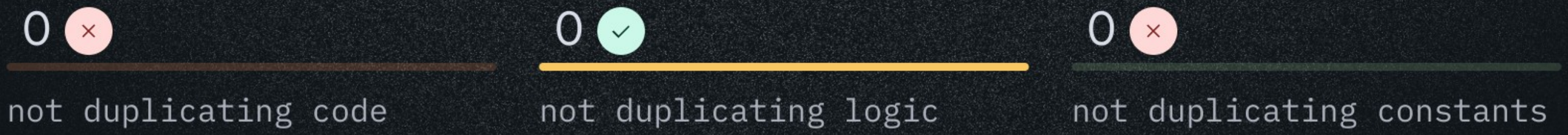
Keeping everything as simple as possible is in 90% case the most important dev strategy

Mentimeter

# The DRY/DIE Principle

Mentimeter

# DRY is about



Mentimeter

# DRY benefits

0 responses

Mentimeter

# DRY benefits

- Maintainability
- Readability
- Efficiency
- Consistency
- Reduced Bugs

Mentimeter

# DRY effects

- Readability and Simplicity
- Abstraction
- Coupling
- Optimizing
- Flexibility

Mentimeter

# Conclusion

“Don’t Repeat Yourself” (DRY) approach promotes readability, maintainability, and code efficiency

Mentimeter

# Q&A part

**0 questions**
**0 upvotes**