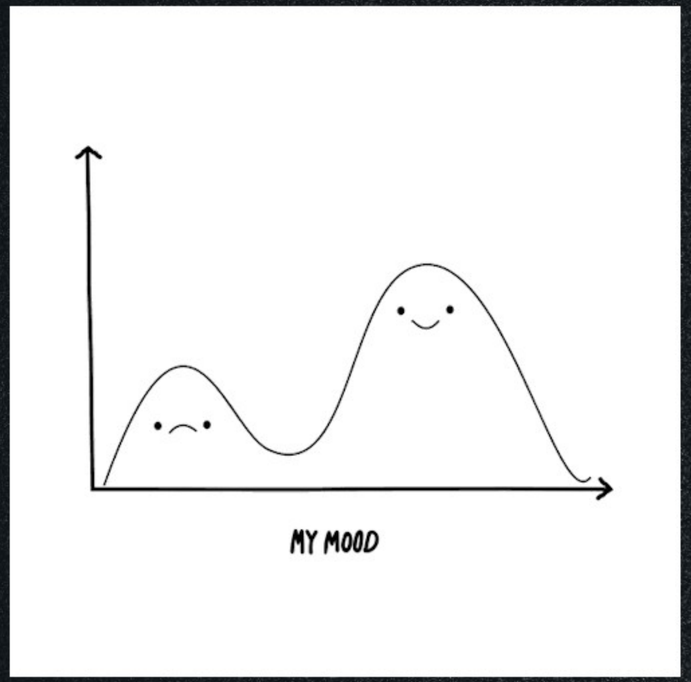
Mentimeter

Instructions

How is it going?

0 responses



Do you have problems with practice?

Agenda

- Recall last session
- Discuss current topic
- Practice

Total Recall

What do you remember most from the previous session?

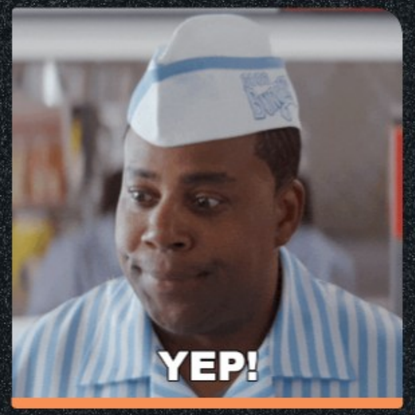
Discuss current topic

Random fact

Matcha also contains a compound called L-theanine, which alters the effects of caffeine, promoting alertness and helping avoid the crash in energy levels that can follow caffeine consumption

Have you looked into DIP and IOC + DI?

0



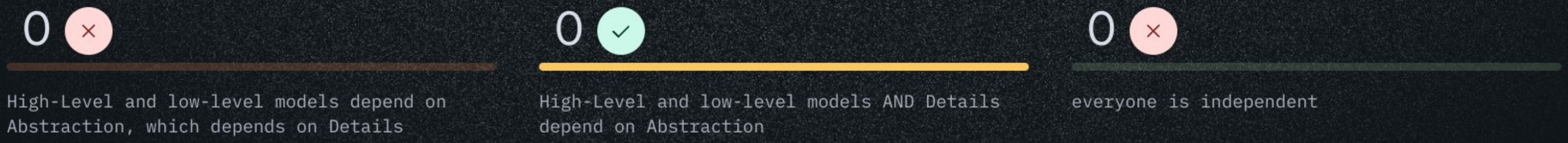
yes

0

no

DIP is about

DIP is about



DIP by Martin

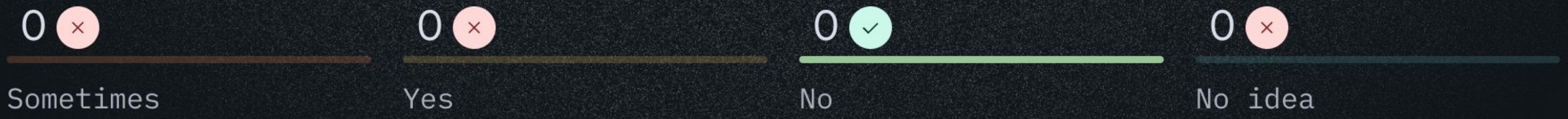
High-level modules should not depend on low-level modules. Both should depend on abstractions.

Abstractions should not depend on details. Details should depend on abstractions.

Abstractions should not depend on details. Details should depend on abstractions. How do you understand this? What are those details?

Mentimeter

DIP is the same as Dependency Injection using Inversion of Control



Basic rule for DIP

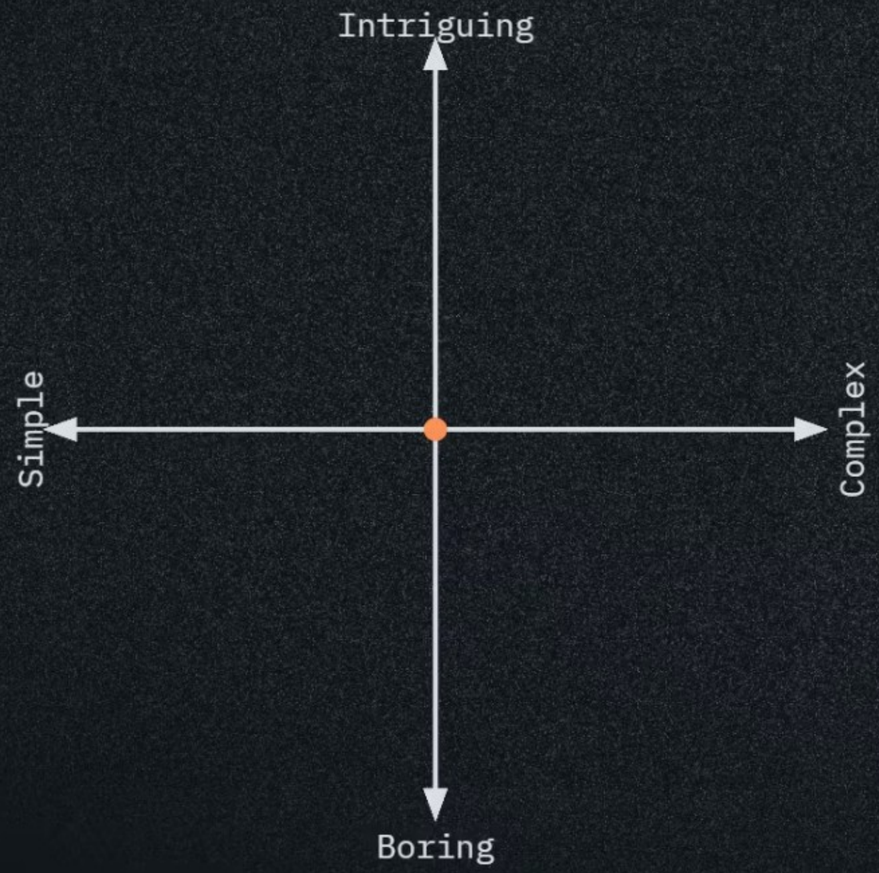
Separate Layers

Learn article

The Dependency Inversion Principle in Java

https://www.baeldung.com/java-dependency-inversion-principle

Express your opinion



1 What do you think about current course

Mentimeter

10 minute break

IoC is

IoC states

that objects do not create other objects on which they depend. Instead, they get the objects they need from an external source (the IoC container).

What is IoC Container?

0 responses

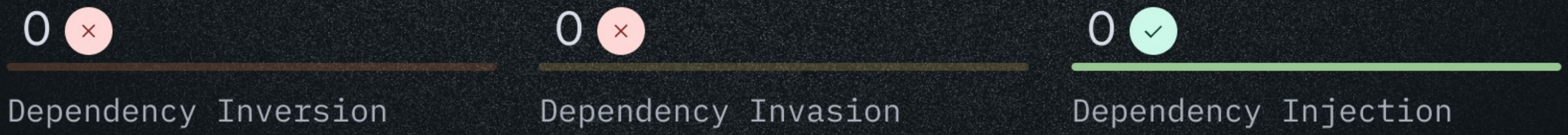
IoC advantages

0 responses

IoC types

- Dependency Injection
- Dependency Lookup

DI is known as



DI advantages

0 responses

Dependency Inversion Principle (DIP)

A *principle* that states that entities should depend on abstractions, not concretions. This allows for high-level and low-level modules to be independent of each other.

Inversion of Control (IoC)

A *principle* that involves shifting the control of object creation and dependency management to a third party, such as a framework or container. This makes it easier to change the implementation of components without changing the components that use them.

We can achieve Inversion of Control through various mechanisms such as: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI).

Dependency Injection (DI)

A *technique* that involves making dependencies available to the user of those dependencies. DI is a *specific type of IoC* that is commonly used to manage dependencies between classes.

Conclusion

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)

Q&A part

0 questions
0 upvotes