

**no SOLID evidence**

**...how to take  
a 5 letter acronym  
no one can use  
and end with  
two phrases you can**

hey hi hello

derek graham

@deejaygraham

A wide-angle photograph of a modern office complex under a blue sky with scattered white clouds. The central building features a large glass facade with a triangular pattern and a grey metal frame. To its left and right are smaller buildings with similar architectural features. The foreground is a grassy field with a paved path leading towards the building. A bright green watermark with the word "sage" in lowercase is overlaid in the lower center of the image.

sage





**NET  
CBYTES**

# Notifications



All

Mentions



**Brian Marick and Kent Beck** Retweeted your Tweet

very excited for tonight's event - @GeePawHill will be with us discussing  
"Steps and Value in Change-Harvesting"

Details here: [ti.to/ne-bytes/june-...](https://ti.to/ne-bytes/june-...) [pic.twitter.com/lXrGuObTdj](https://pic.twitter.com/lXrGuObTdj)

**SOLID?**

# OLD MAN YELLS AT CLOUD



בְּרִיתָהָאַתְּ. אֶלְעָזָרְבָּן  
[ זְנָתָן : מְנֻחָה ]

BEEN THERE.DONE THAT  
CAN'T REMEMBER

123

SESAME STREET.



**Code is Hard**



# Essential & Accidental

ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS



ESSAYS ON SOFTWARE ENGINEERING  
THE  
MYTHICAL  
MAN-MONTH

FREDERICK P. BROOKS, JR.

ESSAYS ON SOFTWARE ENGINEERING

**Wat?**



Robert C. Martin Series

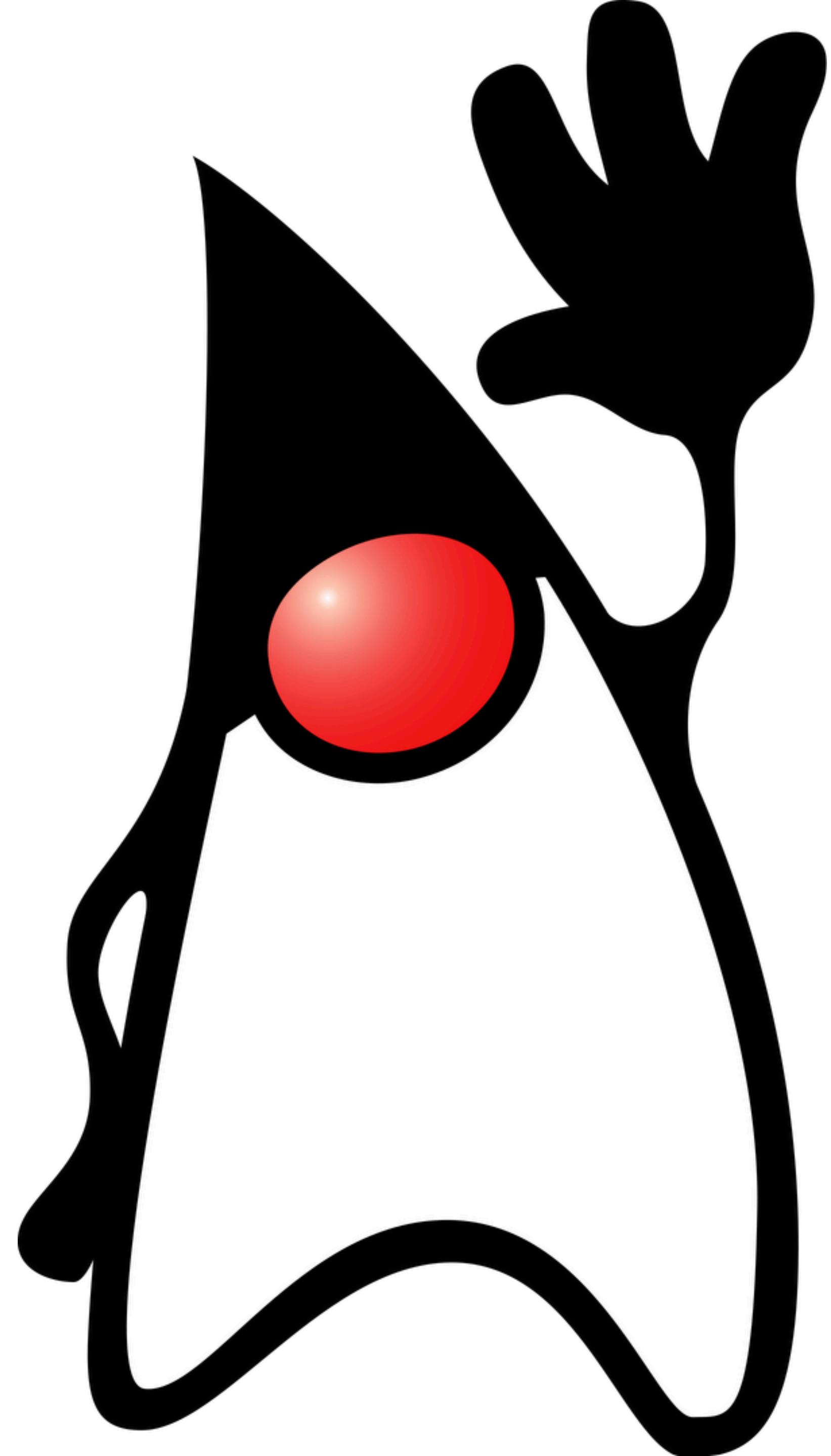
• HALL  
• TAYLOR

# Clean Code

A Handbook of Agile Software Craftsmanship

Foreword by James O. Coplien

Robert C. Martin



**SOLID, dude!**

- Single Responsibility
- Open Closed
- Liskov Substitution
- Interface Segregation
- Depedency Inversion

S O L I D

**Each item of code should have  
a single responsibility**

**“...one reason to change”**

**Each software module has one,  
and only one, reason to change**

- Single Responsibility Cohesion
- Open Closed
- Liskov Substitution
- Interface Segregation
- Depedency Inversion

S O L I D

**Open for extension but  
Closed for modification**

A close-up portrait of Robert C. Martin, an elderly man with grey hair and brown eyes, wearing a light-colored jacket over a dark shirt.

# Object-Oriented Software Construction

A close-up portrait of Robert C. Martin, a middle-aged man with dark, wavy hair and brown eyes, looking directly at the camera with a slight smile. He is wearing a light-colored, textured jacket over a dark shirt.

# **5 Principles of Software Construction**

## **Linguistic Modular Units**

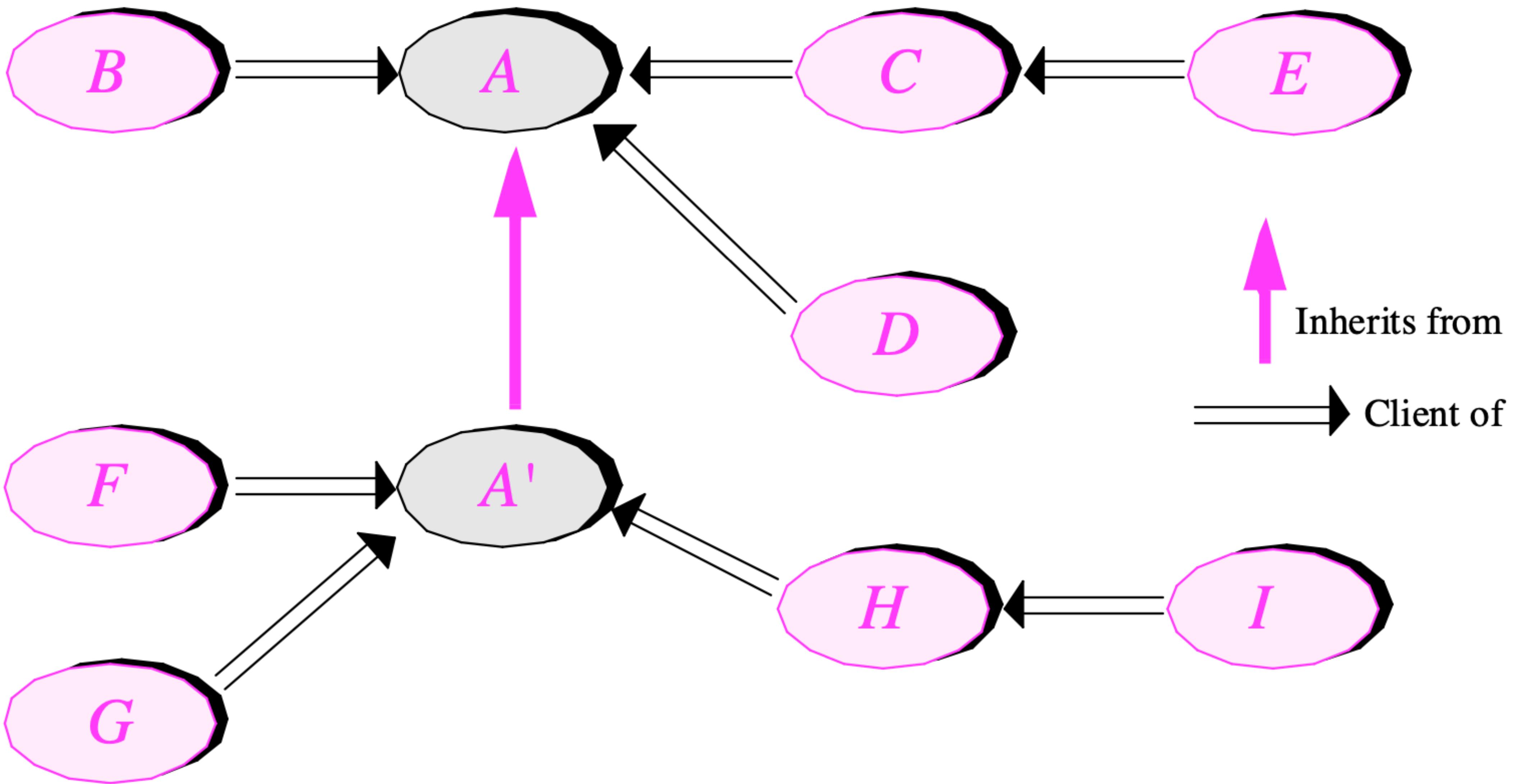
## **Self-Documentation**

## **Uniform Access**

## **Open-Closed**

## **Single Choice**

- A module is said to be open if it is still available for extension. For example, it should be possible to expand its set of operations or add fields to its data structures.
- A module is said to be closed if it is available for use by other modules. This assumes that the module has been given a well-defined, stable description (its interface in the sense of information hiding). At the implementation level, closure for a module also implies that you may compile it, perhaps store it in a library, and make it available for others (its *clients*) to use. In the case of a design or specification module, closing a module simply means having it approved by management, adding it to the project's official repository of accepted software items (often called the project *baseline*), and publishing its interface for the benefit of other module authors.



- Single Responsibility Cohesion
- Open Closed Coupling
- Liskov Substitution
- Interface Segregation
- Depedency Inversion

S O L I D

One subclass can be swapped  
out for another

# Liskov



# Liskov

# Liskov

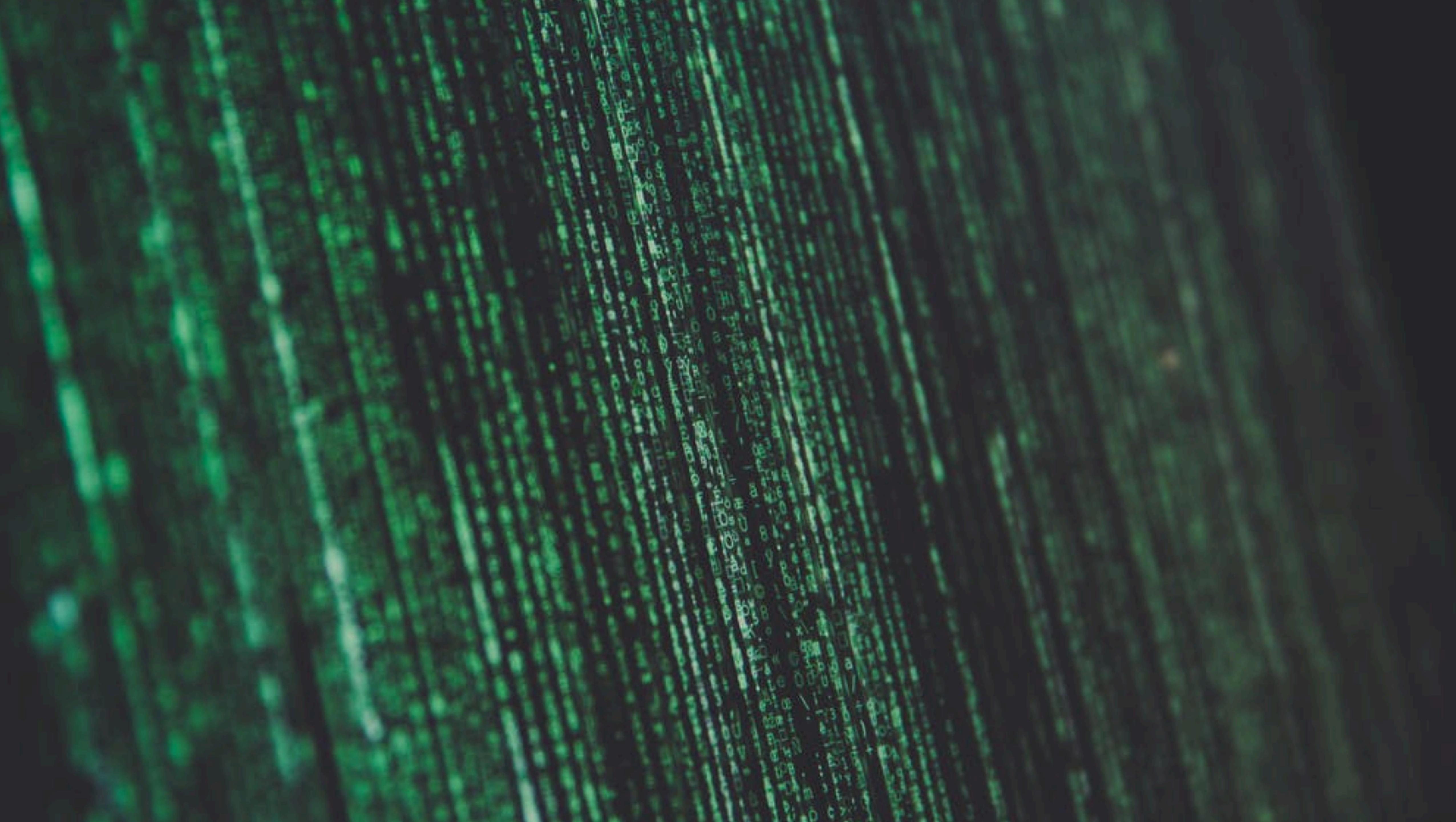


# Liskov

## IEEE Von Neumann

## ACM Turing





- Single Responsibility Cohesion
- Open Closed Coupling
- Liskov Substitution Coupling
- Interface Segregation
- Depedency Inversion

S O L I D

**Many small, specific interfaces are better  
than one general-purpose interface**

- Single Responsibility Cohesion
- Open Closed Coupling
- Liskov Substitution Coupling
- Interface Segregation Cohesion
- Depedency Inversion

S O L I D

**Don't depend on concrete types,  
depend on abstractions**

- Single Responsibility Cohesion
- Open Closed Coupling
- Liskov Substitution Coupling
- Interface Segregation Cohesion
- Dependency Inversion Coupling

# Connascence

**... a change in one would require the other to be modified in order to maintain the overall correctness of the system**

Dynamic:



Static:



# Simple Design

- Appropriate
- Factored
- Communicative
- Minimal

- Tests
- Duplication
- Clarity
- Smallest

- Tests
- Duplication
- Clarity
- Smallest

- Tests
- Duplication
- Clarity
- Smallest

So...

- Single Responsibility
- Open Closed
- Liskov Substitution
- Interface Segregation
- Depedency Inversion

- Single Responsibility Cohesion
- Open Closed Coupling
- Liskov Substitution Coupling
- Interface Segregation Cohesion
- Dependency Inversion Coupling

- Cohesion
- Coupling
- Connascence

- Tests
- Duplication
- Clarity
- Smallest

- Tests
- Duplication
- Clarity
- Smallest

Notice and  
Improve the  
Names of  
Things

@deejaygraham



Questions  
**Answers**