Welcome

OOP in Java

PLURALSIGHT

Hello

About me...

- 27+ years in the industry
- 23+ years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese’s Cups!

Agenda

- Origin of Design Patterns
- SOLID Principles – Architecting for the Future
- Factory Design Pattern

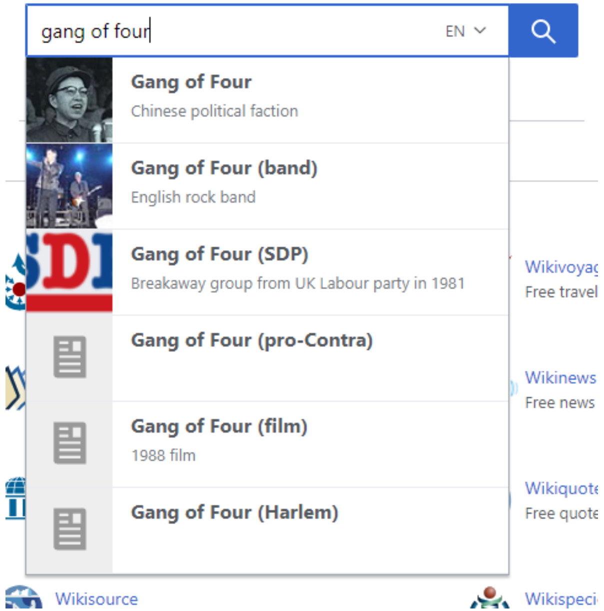
Origin of Design Patterns

First Use of “Design Pattern”

- Term first coined by an architect and anthropologist – Christopher Alexander
- Presented a new language construct based around an entity called a “pattern”
- Pattern describes a problem and provides a reusable (and proven) solution to that problem

Gang of Four (GoF)

- Initial search for “Gang of Four” on www.wikipedia.org



gang of four

EN

Gang of Four
Chinese political faction

Gang of Four (band)
English rock band

Gang of Four (SDP)
Breakaway group from UK Labour party in 1981

Gang of Four (pro-Contra)

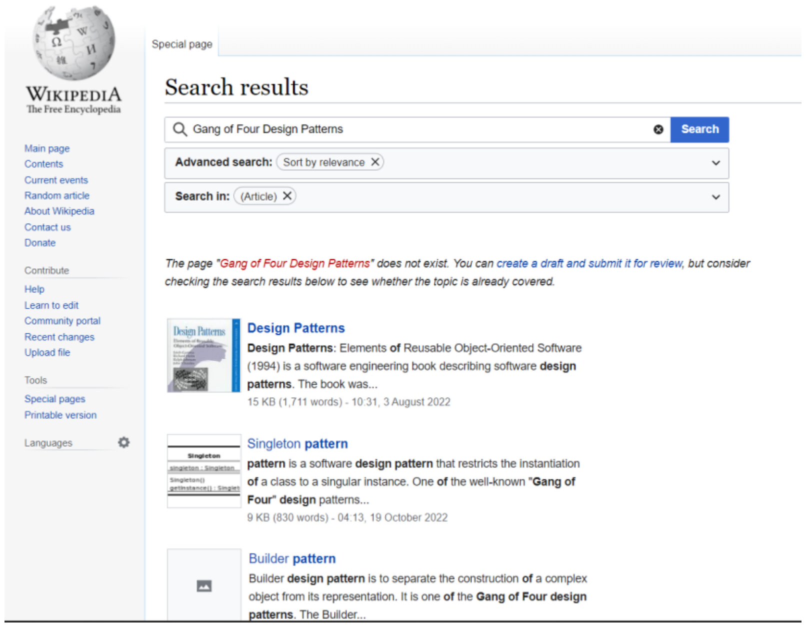
Gang of Four (film)
1988 film

Gang of Four (Harlem)

Not exactly what we’re looking for…

Gang of Four (GoF)

- Search for “Gang of Four Design Patterns”



Search results

Gang of Four Design Patterns

The page "Gang of Four Design Patterns" does not exist. You can create a draft and submit it for review, but consider checking the search results below to see whether the topic is already covered.

Design Patterns
Design Patterns: Elements **of** Reusable Object-Oriented Software (1994) is a software engineering book describing software **design patterns**. The book was...
15 KB (1,711 words) - 10:31, 3 August 2022

Singleton **pattern**
pattern is a software **design pattern** that restricts the instantiation **of** a class to a singular instance. One **of** the well-known **"Gang of Four" design** patterns...
9 KB (830 words) - 04:13, 19 October 2022

Builder **pattern**
Builder **design pattern** is to separate the construction **of** a complex object from its representation. It is one **of** the **Gang of Four design patterns**. The Builder...

That’s more like it…

Gang of Four (GoF)

- Group of 4 authors who wrote the book titled “Design Patterns: Elements of Reusable Object-Oriented Software” (1994)
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
- Includes detail on 3 types of patterns

Gang of Four (GoF)

- Creational
- Structural
- Behavioral

Gang of Four (GoF)

Supports creation of objects indirectly (in a more loosely-coupled fashion); enables association of logic to determine what and how to create

Creational

Structural

Behavioral

Gang of Four (GoF)

Creational

Structural

Behavioral

About class and object composition; using inheritance and extension to build out entity hierarchies that match with the “real world” and enable layering in new functionality in an architecturally sound manner

Gang of Four (GoF)

Creational

Structural

Behavioral

Mainly manages concepts of communication between objects – building out a messaging system that allows us to break a larger problem into smaller pieces but still coordinate

SOLID Principles – Architecting for the Future

SOLID Principles

SOLID principles help us build testable and more maintainable code

- Single Responsibility Principle (SRP)
- Open-Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Single Responsibility Principle (SRP)

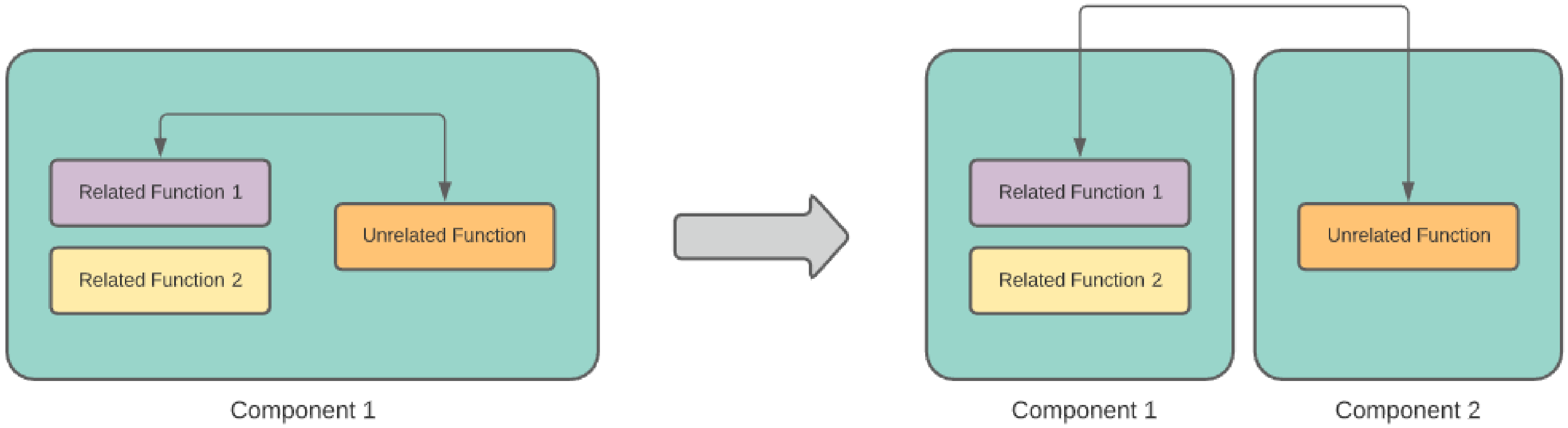
Single Responsibility Principle (SRP)

A system module or component should have only one reason to change

Single Responsibility Principle (SRP)

SINGLE RESPONSIBILITY PRINCIPLE
Every object should have a single responsibility, and all its services should be narrowly aligned with that responsibility.

Single Responsibility Principle (SRP)

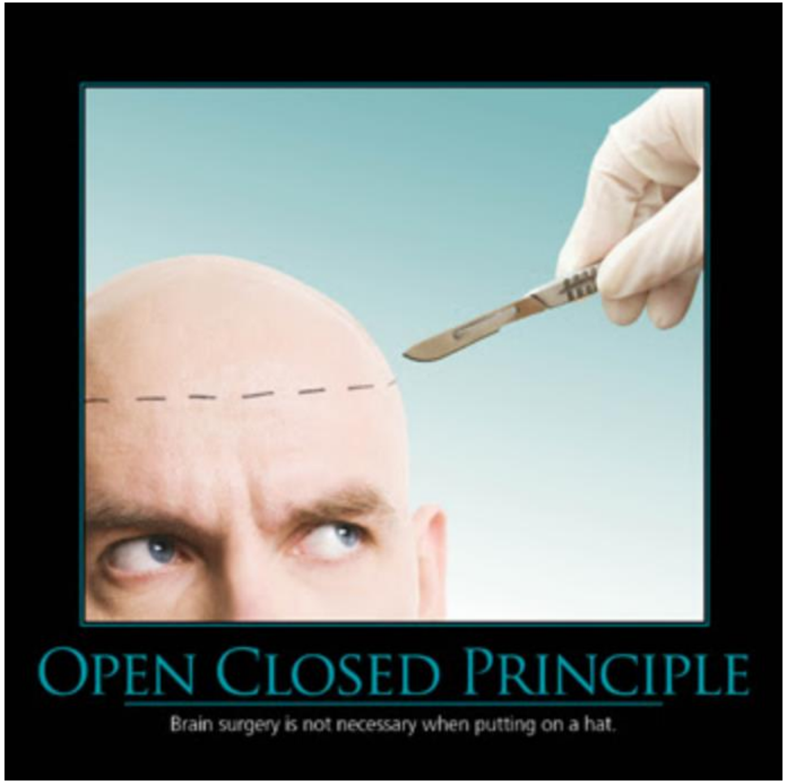


Open-Closed Principle (OCP)

Open-Closed Principle (OCP)

Software entities should be open for extension but closed for modification

Open-Closed Principle (OCP)

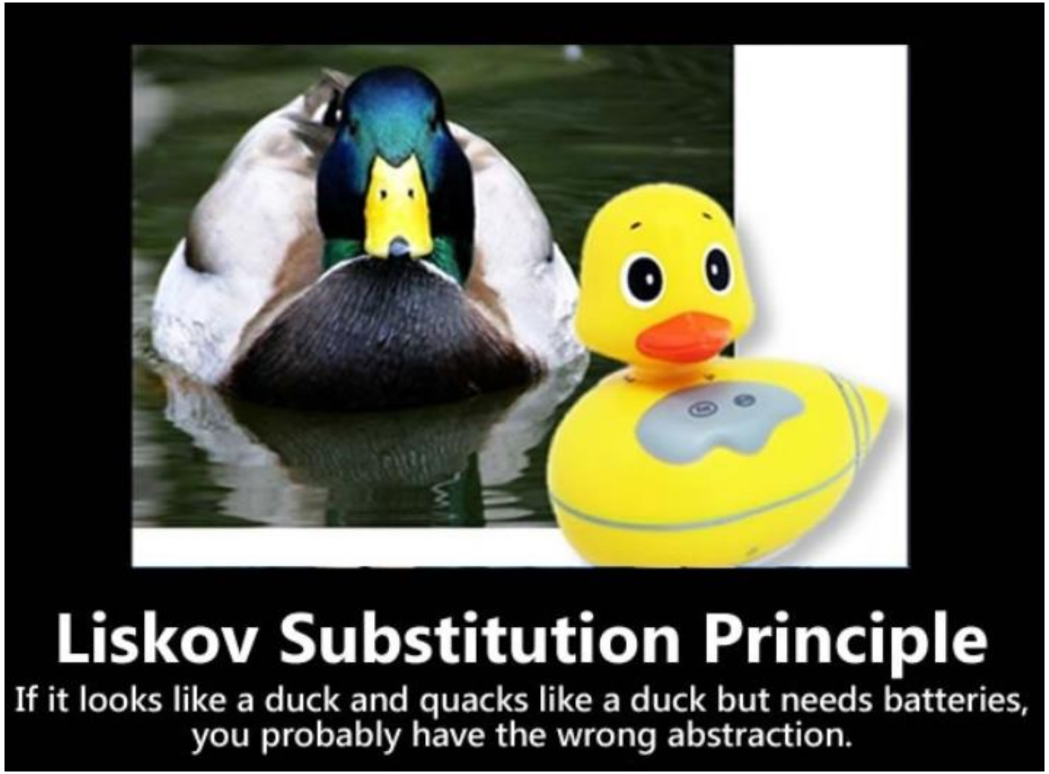


Liskov Substitution Principle (LSP)

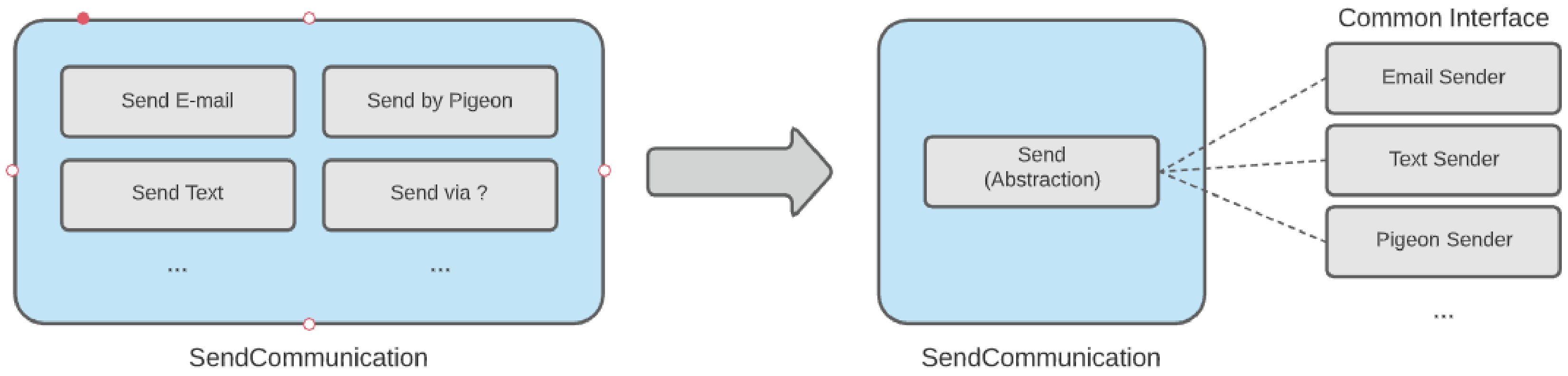
Liskov Substitution Principle (LSP)

Subtypes must be substitutable for their base types

Liskov Substitution Principle (LSP)


Liskov Substitution Principle
If it looks like a duck and quacks like a duck but needs batteries, you probably have the wrong abstraction.

OCP & LSP

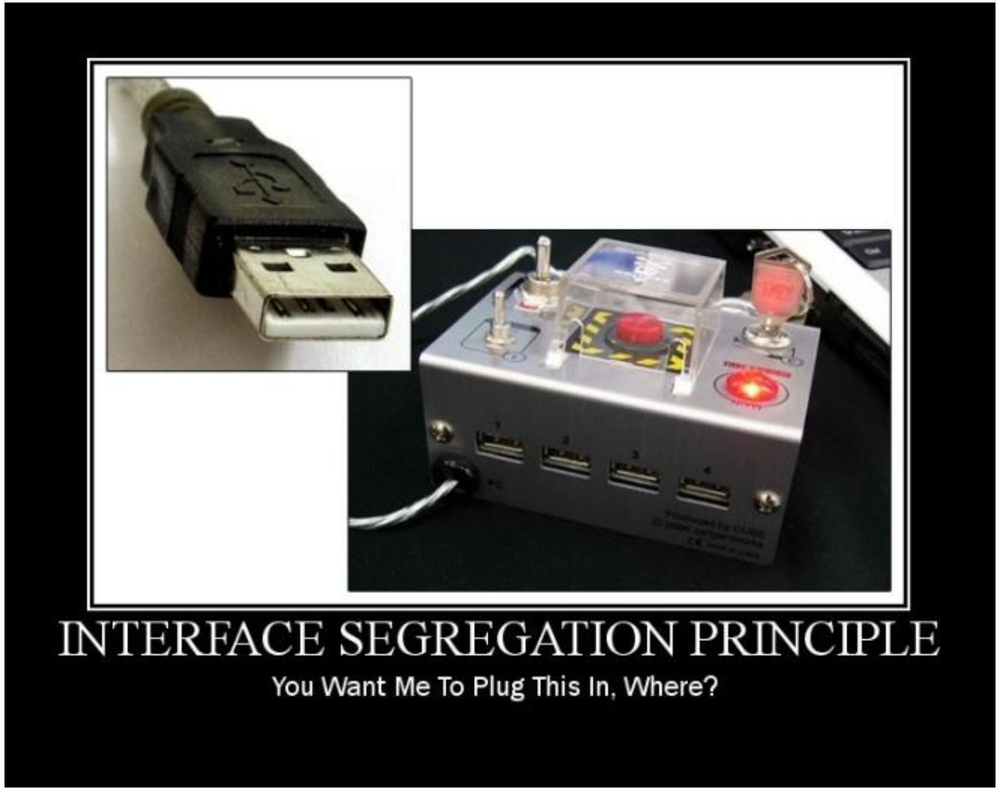

Send E-mail
Send by Pigeon
Send Text
Send via ?
...
...
SendCommunication
Send (Abstraction)
SendCommunication
Common Interface
Email Sender
Text Sender
Pigeon Sender
...

Interface Segregation Principle (ISP)

Interface Segregation Principle (ISP)

Clients should not be forced to depend on methods they do not use

Interface Segregation Principle (ISP)


INTERFACE SEGREGATION PRINCIPLE
You Want Me To Plug This In, Where?

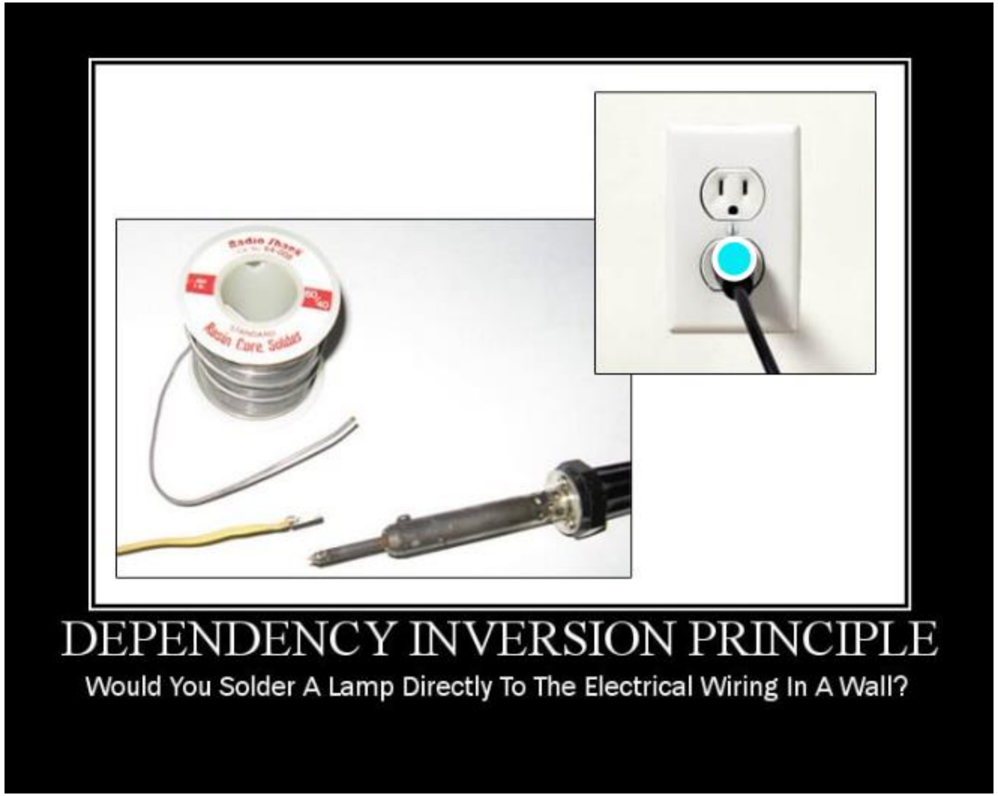
Dependency Inversion Principle (DIP)

Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules – both should depend on abstractions

Abstractions should not depend upon details – details should depend upon abstractions

Dependency Inversion Principle (DIP)


Radio Shack
Rosin Core Solder
DEPENDENCY INVERSION PRINCIPLE
Would You Solder A Lamp Directly To The Electrical Wiring In A Wall?

Design Patterns in Software Architecture

Design Patterns

- As previously discussed, design patterns are proven solutions to a specific technical problem
- Focused primarily on the level of the source code
- Different classes of problem/solution that can be used (and reused) as building blocks

Factory Pattern

- Creational pattern used to abstract creation logic from client
- Rather than create directly, code uses the factory to generate new instances
- Provides way to vary what gets created (and how) based on business logic

Thank you!

If you have additional questions, please reach out to me at: asanders@gamuttechnologysvcs.com

PLURALSIGHT