# Design Principles

# Roadmap for Design Principles

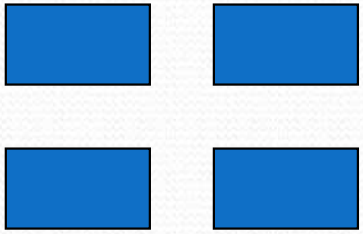
- Cohesion and Coupling
- KISS
- DRY and WET ##

# Coupling and Cohesion

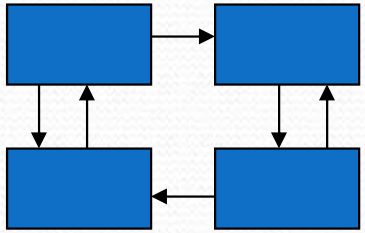
# Characteristics of Good Design

- Component independence
  - High Cohesion
  - Low Coupling

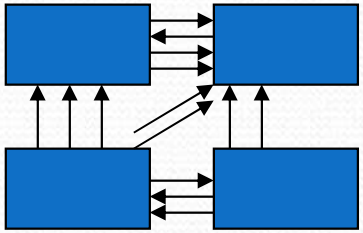
# Coupling: Degree of dependence among components



No dependencies



Loosely coupled-some dependencies



Highly coupled-many dependencies

High coupling makes modifying parts of the system difficult, e.g., modifying a component affects all the components to which the component is connected.

# Cohesion

- Definition: The degree to which all elements of a component are directed towards a single task and all elements directed towards that task are contained in a single component.
- Internal glue with which component is constructed
- All elements of component are directed toward and essential for performing the same task
- High is good ##

# Cohesion and Coupling

- Cohesion is a measure of:
  - functional strength of a module.
  - A cohesive module performs a single task or function.
- Coupling between two modules:
  - a measure of the degree of interdependence or interaction between the two modules. ##

# Cohesion and Coupling

- A module having high cohesion and low coupling:
  - functionally independent of other modules:
    - A functionally independent module has minimal interaction with other modules.

## Advantages of Functional Independence

- Better understandability and good design:
- Complexity of design is reduced,
- Different modules easily understood in isolation:
  - modules are independent

# Advantages of Functional Independence

- Functional independence reduces error propagation.
  - degree of interaction between modules is low.
  - an error existing in one module does not directly affect other modules.
- Reuse of modules is possible.

# Advantages of Functional Independence

- A functionally independent module:
  - can be easily taken out and reused in a different program.
    - each module does some well-defined and precise function
    - the interfaces of a module with other modules is simple and minimal.

# KISS


Naughty minds ☺ ;-)

# K.I.S.S. Design Principle

- Keep It Simple Stupid
- Keep It Short and Simple
- Keep It Simple and Straight Forward ##

# K.I.S.S. Design Principle

# K.I.S.S. Design Principle

- Story of the Soap Factory

# DRY

Do not Repeat Yourself

# Root cause: Needless Repetition

- The design contains repeating structures that could be unified under a single abstraction
- The problem is due to developer's abuse of cut and paste.
- It is really hard to maintain and understand the system with duplicated code.

***Duplication is Evil!***

***DRY – DON'T REPEAT YOURSELF***

# DRY Design Principle

- **Don't Repeat Yourself** (**DRY**) is a principle of software development aimed at reducing repetition of information of all kinds, especially useful in multi-tier architectures
- The principle has been formulated by Andy Hunt and Dave Thomas in their book “The Pragmatic Programmer”
- When the DRY principle is applied successfully, a modification of any single element of a system does not require a change in other logically unrelated elements.

- Violations of DRY are typically referred to as **WET** solutions, which stands for "**Write Everything Twice**".