# Design Principles

Coupling and Cohesion

*Weeks of coding can save you hours of design.*

## Coupling

- Describes the interdependence of entities
- Bound together to be functional

## Coupling

- High/strong coupling
  - One object uses the internal data of another directly
  - Returning a pointer to object data
  - Sharing global variables
  - C++ friends

A high coupling could be risky.

## Coupling

- Medium coupling
  - Controlling the flow of another entity
  - Passing entire data structures when only some data is needed

good!

ideally.

- Low coupling
  - Share only data, through parameters and returns
  - Events or messages passed

well design interfaces.

## Coupling

- Low coupling leads to information hiding and encapsulation
  - Stable interfaces
  - Better maintainability
- High coupling leads to interdependent entities
  - Design is difficult to change and extend
  - Code is difficult to read

## Cohesion

- Intra-relatedness or focus of an entity
- Relatedness of functionality and/or the data of an entity

## Cohesion

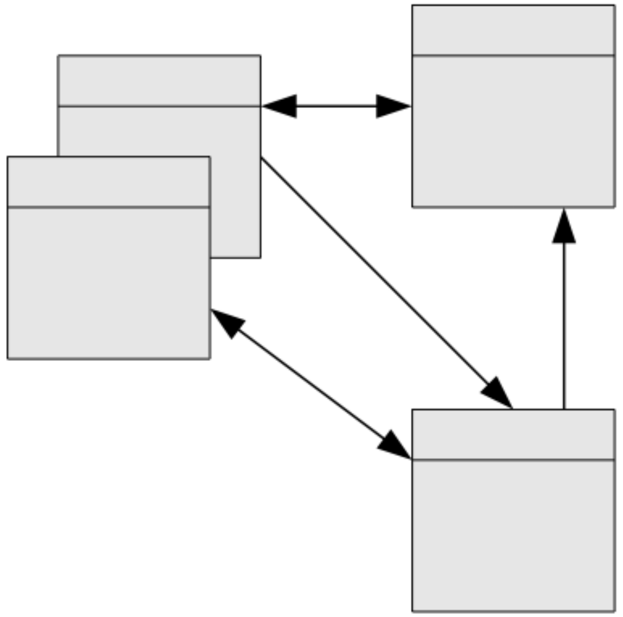
- Low cohesion
  - Related only because they are grouped together
  - Entity names often vague
  - Functions provided perform various activities
  - Groups of functions often act on distinct sets of data

## Cohesion

- High cohesion
    - Entity's members contribute to a well defined purpose
    - Entity name often very specific
    - Functions act on the same set of data
    - Improved clarity of entities and their dependencies
    - Easier to maintain and extend
    - Leads to code reuse

## High Coupling / Low Cohesion

- Hard to read and understand
- Boundaries between entities are weak and few
- High interdependence causes unstable code



# Low Coupling / High Cohesion

- Easier to read, maintain and extend
- Entities are responsible for specific functionality
- Entities function more independently