Software Engineering
Module 2

Program

Giancarlo Succi

Structure

- Portion of the course will be online
- The specific plan of the lectures will be announced as we progress
- The teaching material is in GitHub: https://github.com/GiancarloSucci/UniBo.SE.A2024

Please register yourself for updates etc

- Link: http://tiny.cc/QuadernoIS24

Use of AI

- We will take advantage an experimental AI platform for reviewing material:
  - research.constructor.tech

The knowledge base

- The knowledge base of the course will be populated step-by-step
- Its link is: https://research.constructor.tech/platform/knowledge-models/44814c2adf2c42d9958c5de096336237/chat

Remember the grading strategy

- Let O be the grade of the oral
- If O < 18 -> Fail
- Else let O = 18 + (O-18)/4
- Let P be the grade of the optional project 0 <= P <= 10
- Let G = P + O
- If G > 30 -> 30 cum Laude, else G

Notice that at the first session the oral can be substituted by a written test

Program of the module

- Premises
- Introduction to UML
- Design Patterns
- Architectures
- Contextualization in Python / ChatGPT

Premises

- Sub-system and module
- Information hiding
- Coupling
- Cohesion
- Simplicity

Sub-system and module

- Sub-system
  - Performing specific task or subset of responsibilities (procedural approach)
  - Set of classes (OO approach)
  - Example: DBMS and error processing system
- Module
  - More language-specific
  - Set or library of functions performing specific tasks (procedural approach)
  - Classes (OO approach)

Information hiding

- Module must hide its internal implementation
- Module accessed only through public interface
  - No direct access to internal data & private methods
  - Data accessed through a well defined set of accessor methods
- Use abstraction to define modules & interfaces
- Changing implementation (given no change to interface) should have no effect on rest of system

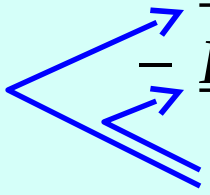
Low coupling

- Two modules are loosely coupled
  - If interconnections and dependencies are weak
  - Satisfying info hiding better than high coupling
- (Increasing) coupling order
  - Methods of a module calling another method's
  - Data coupling/control coupling
  - Class of a module is a subclass of another module's
  - Module(s) making use of specific features of compiler or calls to specific API procedures of the OS
  - I/O coupling
  - Common coupling
  - Content coupling

Avoid!

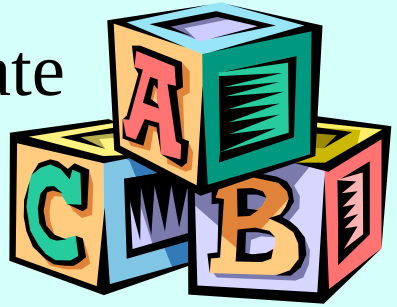
Cohesion

- Cohesive module: all its elements directed toward performing a single task
- Increasing magnitude of cohesion
  - *Coincidental*: parts grouped together for no reason
  - *Logical*: logically related parts, no other interactions
  - *Temporal*: parts processed within same time limit
  - *Procedural*: control flows from one part to another
  - *Communicational*: parts related by same I/O
  - *Sequential*: output of one is the input of another
  - *Informational*: access to same data structure
  - *Functional*: all elements for one single concept



Best to achieve

Simplicity



- Build only needed code; don't try to anticipate future needs
- Refactoring
  - Restructuring a working system to make it simpler
- Simplicity at different levels
  - Method level: short methods with small signatures
  - Module level: small public interface
  - System level:
    - Avoid "middle-man" modules and global variables
    - Minimize info and control paths
    - Keep inheritance hierarchies small
  - At all level: avoid code duplications