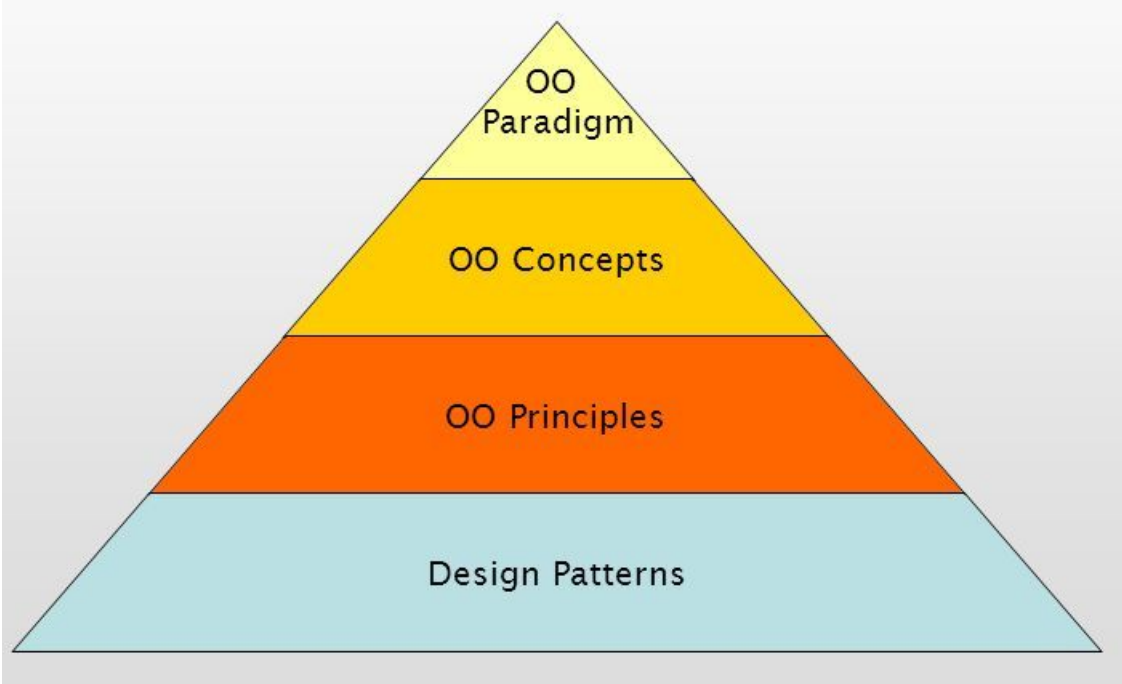
C++ Programming

Pyramid of Object Oriented

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

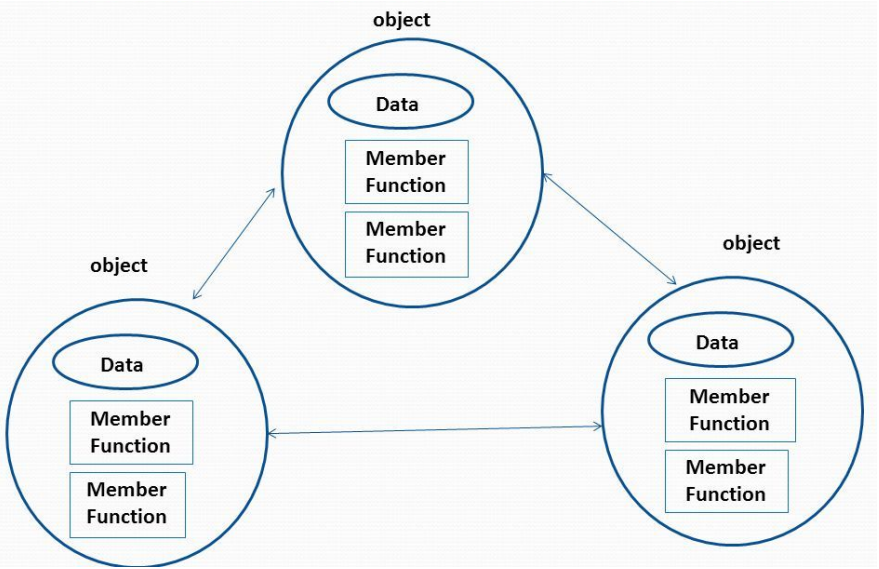
The pyramid of OO (Object Oriented)



Img Src

OO Programing Paradigm

- Programing paradigm: Way of thinking/viewing/**structuring** for a software
- **OOP is a programing paradigm**
  - View: Objects + Functions + Interactions
  - Very centralized around object concept
- Procedural programming is another one
  - Bunch of files and functions + linear flow of instructions
- Other paradigm
  - Declarative, Functional, etc

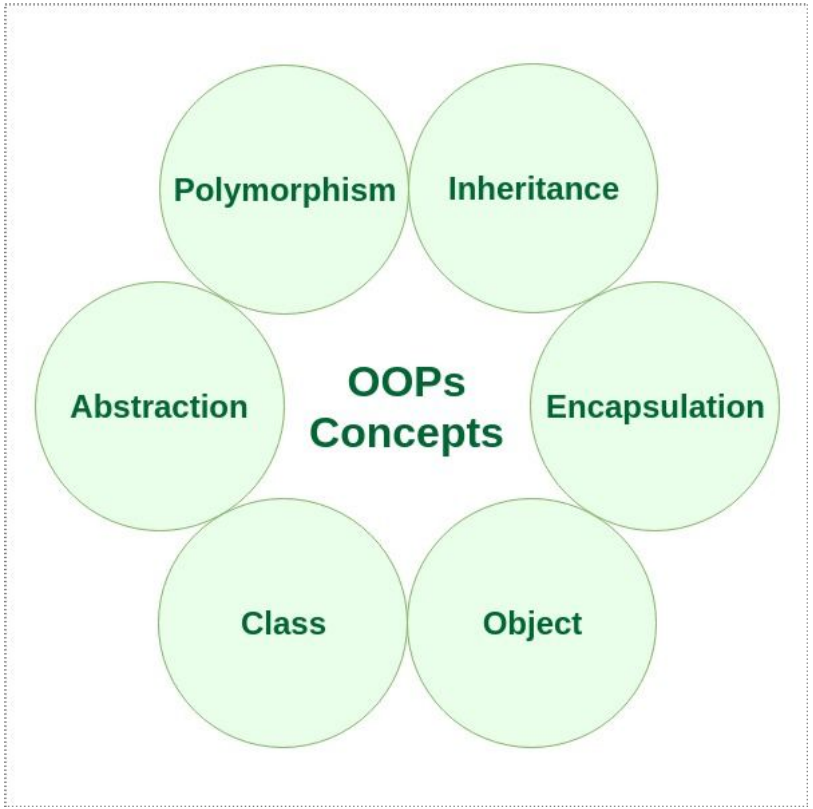


Img Src

OO Programing Paradigm: WHY!

- Close **correspondence** between real-world objects and OOP classes.
- Help in Handling complexity of software products
- Seems one of best ways to deal with Software Crisis
  - Complex projects ⇒ Over-budget, over-time, buggy, **!**meet requirements, never delivered
- OOP languages are good to an extent for handling complex projects
- On the other side
  - Some people criticize a lot OOP itself
  - Others criticize the current designs/focus of current OOP relative the old intentions
- Why not procedural?
  - No owner for data / data integrity issue / Many functions may modify the building block
  - Harder debugging if data is corrupted

OO Concepts



Img Src

OO Principles

- The most important skills we need in design!
- SOLID Principles
  - Single Responsibility Principle (SRP)
  - Open/Closed Principle (OCP)
  - Liskov Substitution Principle (LSP)
  - Interface Segregation Principle (ISP)
  - Dependency Inversion Principle (DIP)
- DRY (Don’t Repeat Yourself)
- KISS (Keep it simple, Stupid!)
- YAGNI (You ain't gonna need it)
- *Several design principles will be embedded implicitly in the homework*

Design Patterns

- The **best practices** for some **repetitive** design sub-tasks
- Fatal Mistakes: Overstress in study & Overuse in projects
- Skill: Use it in the right situation for the right reasons
- Some patterns are
  - so important (e.g. Singleton / Factor)
  - others are less faced
- From a **domain** to another, some patterns are more used
- *Several design patterns will be embedded implicitly in the homework*

OO**A**, OO**D**, OO**P**

- Let’s say we have **customer requirements** for a specific product
- OOA is an **analysis** phase to these requirements
  - Output: analysis models (use cases & object conceptual model - technology **independent**)
- After software analysis, we **design** the system (OOD)
  - Considers: hardware and software platform, availability, scalability, budget, etc
  - Designing is a skill. It **takes time** to build elegant designs
- Then, we **implement** & test the system, using an OOP language
  - We coding a specific OOP language
- Company Culture + Scale of project + team size ⇒ Decide how the 3 are applied
  - **Small** projects: all of that can be done in a unified way by a small team
  - **Large** projects may have: business analyst, system analyst, architects, tech leads and devs

Misc

- Reading
  - OOD: After course read: *Head First Object-Oriented Design and Analysis* book
  - More in future / Also Designing Data-Intensive Applications book
- Coding Style
  - https://github.com/isocpp/CppCoreGuidelines
  - https://google.github.io/styleguide/cppguide.html

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."