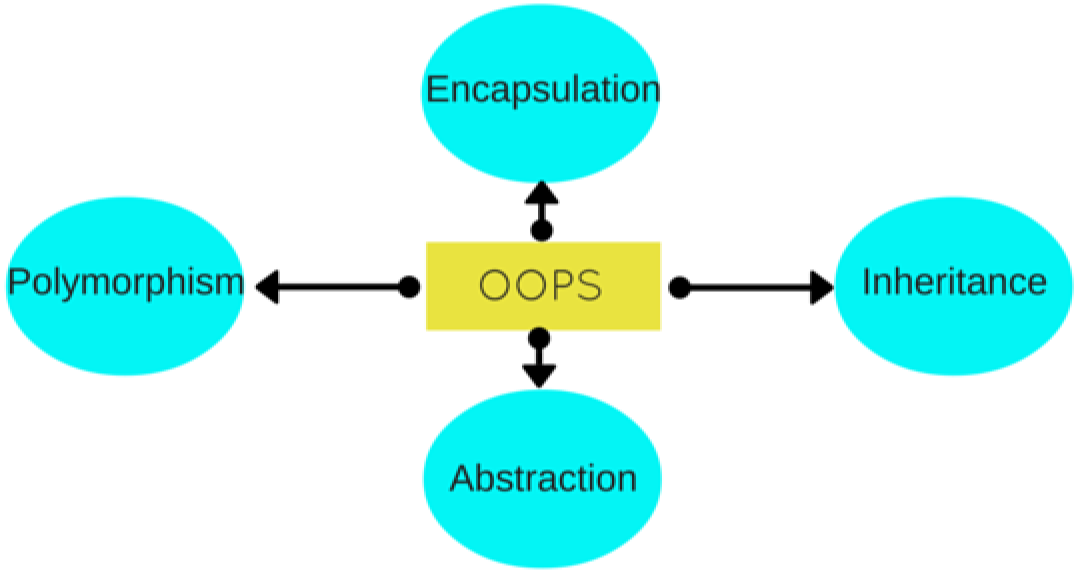
ECE 364
Software Engineering Tools Laboratory

Lecture 7A

Object Oriented Programming Concepts

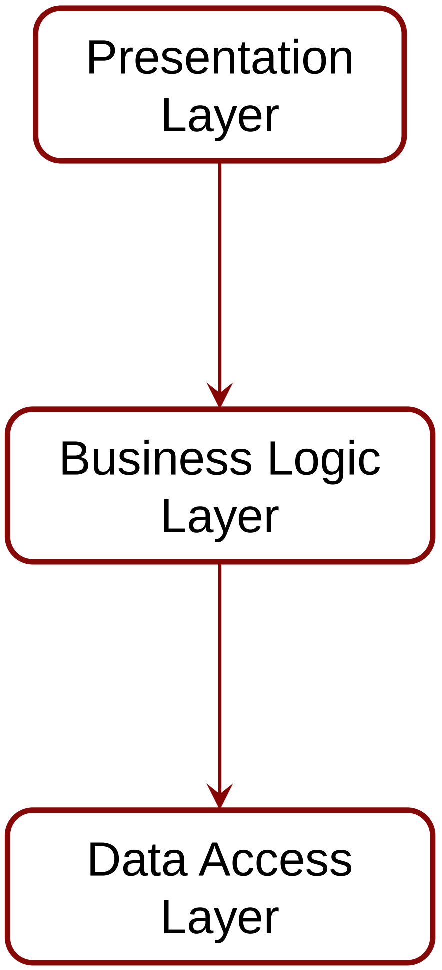
The Concepts



SOLID Principles

- Single Responsibility.
  - Every class should have a single responsibility entirely encapsulated by that class.
  - There should only be a single reason for making a change to the class.
- Open/Closed.
  - Apps should be open for extension but closed for modification.
  - Should be easily extensible but should not need any changes to the core implementations.
- Liskov Substitution.
  - Given a class hierarchy, derived classes should be substitutable for their base classes.
- Interface Segregation.
  - Many small interfaces are better than one general-purpose interface.
  - Classes should NOT be forced to implement members it does not use.
- Dependency Inversion. (Inversion of Control)
  - High-level modules should not depend on low-level modules. Both should depend on abstractions.
  - Abstractions should not depend on details. Details should depend on abstractions.

N-Tier Architecture



Front layer is responsible for “presenting” the data to the user AND acquiring/managing the user input.

The BL layer contains the core business requirements that consumes the data, but drives the UI.

Backend layer is responsible for accessing data stores, and outside resources, mostly, the database.

Design Patterns

- Reusable Solutions for Common Problems.
- Follow different classification methods.
- Most common classification is the GOF:
  - Creational Patterns: Controls the creation of objects.
  - Structural Patterns: Defines the structure between multiple objects.
  - Behavioral Patterns: Defines communication between multiple objects.
  - Concurrency Patterns
- Architectural Patterns are usually for large scale solutions.
  - Model-View Controller (MVC)
  - Repository
  - Extract-Transform-Load (ETL)

Design Patterns 2

Singleton Pattern

- Ensures that only ONE instance exists in memory.
- Most often uses a private or a protected constructor.
- Provides common access point to all consumers.
- Commonly used by other creational patterns.
- Used for state/resource monitoring and capture.
  - How many objects accessed a resource.
  - What is the state of some data after several logical steps.

Design Patterns 3

Model-View-Controller

- This is primarily used for the presentation layer.
- Model:
  - Holds the data, and sometimes, some minimal internal data processing logic.
- View:
  - Only knows what model to expect.
  - Only contains display logic.
  - Must NOT contain data processing.
- Controller:
  - Populates the model from somewhere: Internally, from the DB, from a web call, etc.
  - Processes the model to make it ready to be displayed.
  - Sends the model to the view.

Design Pattern 4

Factory Method:

- A Creational Pattern from the GOF.
- Defines an interface for creating an object.
- Factory Method lets a class defer instantiation to subclasses to decide.
- Common Points:
  - There are usually two class hierarchies: creators, created.
  - The creator class is usually abstract.
  - The created class hierarchy is not required, but simplifies implementation.