Week 9 Design Concepts

- **Software Design**
  - Definition
    - Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation
- **Design vs Analysis**
  - Analysis
    - Identifies "what" the system must do
    - Emphasize an investigation of the problem and requirements
    - The analyst seeks to understand the organization, its requirements nad its objectives
      - Do the right thing
  - Design
    - Specifies "how" it will do it
    - Emphasizes a conceptual solution that fulfills the requirements rather than its implementation
    - The designer seeks to specify a system that will fit the organization, provide its requirements effectively and assist it to meet its objectives
      - Do the thing right
- **Software Design Model**
  - Definition
    - Design modeling in software engineering represents the features of the software that helps engineers to develop it effectively
  - Classification
    - ❶ Data design / class design
      - Transforms class models into design class realizations and the requisite data structure required to implement it
    - ❷ Architectural design
      - Defines the relationship between major structural elements of the software
    - ❸ Interface design
      - Describes how the software communicates with systems that interoperate with it, and with humans who use it
    - ❹ Component-level design
      - Transforms structural elements of the software architecture into a procedural description of software components
- **Software Design Quality (FURPS)**
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
    - Extensibility
    - Adaptability
    - Serviceability
- **Concepts and Principles**
  - Abstraction
    - Hiding the details to reduce complexity and increases efficiency or quality

- Modularity
  - Modularization criteria
    - Coupling
    - Cohesion
- Coupling
  - Definition
    - Coupling refers to how focused a class or a module is
  - Advantage
    - Coupling increases with the complexity and obscurity of the interface between modules
  - Dependency
    - Two modules are considered independent if one can function completely without the presence of the other -- so that they are solvable and modifiable separately
    - The mode connections between the modules, the more dependent they are
  - Highly (tight) coupled
    - Modules are joined by strong interconnections
  - Loosely coupled
    - Weak interconnections
  - Lower coupling
    - Minimize the number of interfaces per module and the complexity of each interface
    - Strengthen the bond between elements of the same module by maximizing the relationship between elements of the same module
- Cohesion
  - Definition
    - The degree of how closely the elements of a module are related to each other
    - A single unit represents a single part of the problem solution
  - Advantage
    - Gives the designer the idea about whether the different elements of a module belong together in the same module
  - Three aspects
    - Method cohesion
      - Clearly defined, all statements in the method contribute to implemeting this function
    - Class cohesion
      - Implement a single concept or abstraction with all elements contributing toward supporting this concept
    - Inheritance cohesion
      - The inheritance promotes reusability but reduce cohesion.
- Object-oriented design
  - An object-oriented system is made of interacting objects that maintain their own local state and provide operations on that state