Engineering of Software Subsystems
Design Principles

```
Component
  operation

Concrete Component
  operation()

Concrete Dec #1
  addedState
  operation()

Decorator
  operation()

Component->operation()

Concrete Dec #2
  addedBehavior()
  operation()

Decorator::operation()
  addedBehavior()
```

Diagram:

- Component
  - operation
    - Concrete Component
      - operation()
        - Concrete Dec #1
          - addedState
            - operation()
        - Concrete Dec #2
          - addedBehavior()
            - operation()
      - Decorator
        - operation()
          - component
            - component->operation()
The object-oriented paradigm is considered to be based on four principles.

- **Encapsulation**
  - *State or attributes*
  - *Behaviors that act on or use the encapsulated data*

- **Inheritance**

- **Polymorphism**

- **Identity**
  - *There are individually identifiable objects that are instantiations of a class.*
There are some key object-oriented design concepts that we will stress.

- Increase cohesion where possible
- Decrease coupling where possible
- Behaviors follow data
- Prefer type (interface) inheritance over class (implementation) inheritance. “Program to the interface, not the implementation.”
- Prefer composition to inheritance: prefer “has-a” relationships to “is-a” relationships.
- Use delegation to “simulate” runtime inheritance.
- Law of Demeter “Only talk to your friends.”
More O-O Design Concepts

- **Abstraction**: provides well defined conceptual boundaries that focus on the outside view of an object and so serves to separate an object’s essential behavior from its implementation.

- **Principle of Least Commitment**: the interface of an object provides its essential behavior, and nothing more.

- **Principle of Least Astonishment**: an abstraction captures the entire behavior of an object and offers no surprises or side effects that go beyond the scope of the abstraction.

- Abstraction is one of the fundamental ways to deal with complexity and complexity’s close cousin – *change*. 
More O-O Design Concepts

- **Open-Closed Principle (OCP)** – software entities (classes, modules, etc.) should be *open* for extension, but *closed* for modification.
- We should attempt to design modules that never need to be changed.
- To extend the behavior of a system, we add new code. We do not modify old code.