

# **CS 407**

## **Design Patterns**

### **Composition and Decorator**

Dr. Chad Williams  
Central Connecticut State University

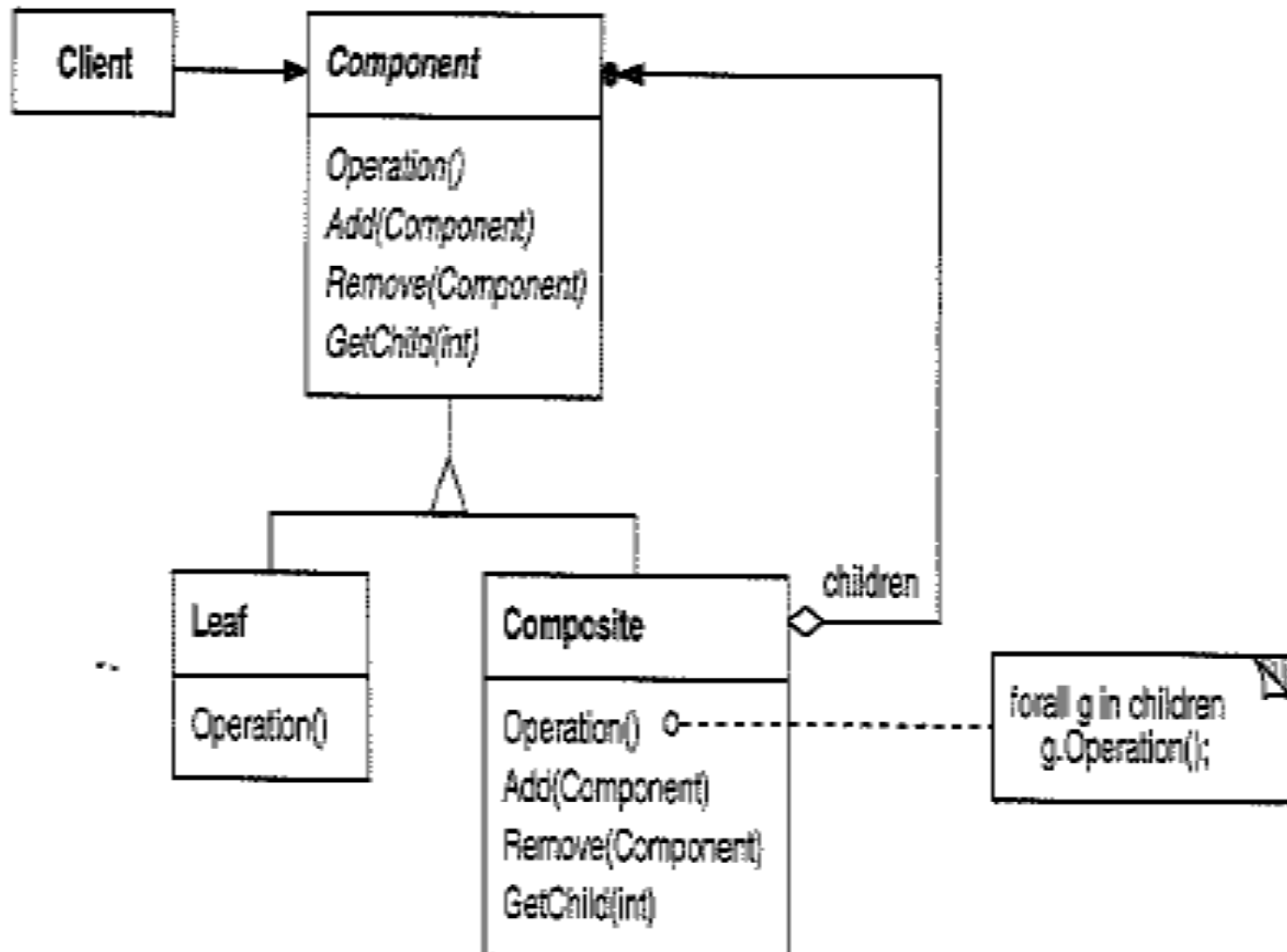
# Design pattern: Composite

- Structural pattern
- Motivation:
  - Group smaller components together and treat as a single large component. Referred to as part-whole hierarchies of objects.
  - Client able to ignore whether interacting with single object or group of objects
- Example:

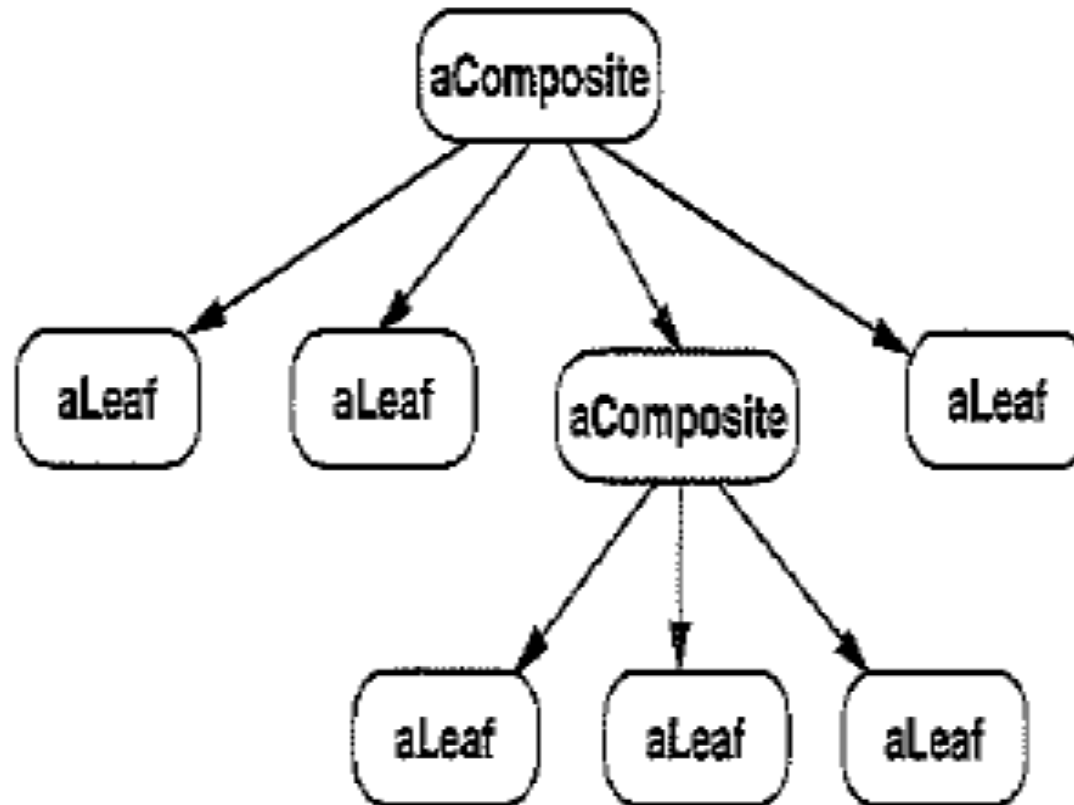
Application window – tell application to draw graphics

  - Menu bar
    - File menu, Home, Insert,...
      - File->Save, Save as, Open,...
  - Toolbar
    - Set of options bold, underline, some are menus

# Composite UML



# Sample instance structure



# Considerations

- Parents?
- Want to treat all components the same without loss of functionality
  - What about how to deal with children?
    - Wait! We talked about how shouldn't inherit from a class if didn't implement all functionality
      - Trade-off of safety and transparency – different approaches depending on application
    - Option 1: Try to make meaningful
    - Option 2: Composite method
    - Option 3: Defined exception

# Relevant patterns

- Sorting of children common – example GUI where consider which element is in front of another
- Iterator over children – usually immediate children, but could have scenario where you would want deep iteration
- Frequently combined with Decorator pattern
- Visitor – localizes operations and behaviors (may get to this later)

# Design pattern: Decorator

- Structural pattern
- Motivation:
  - Attach additional responsibilities dynamically
  - Additional flexibility compared to extensive subclassing
- Example:

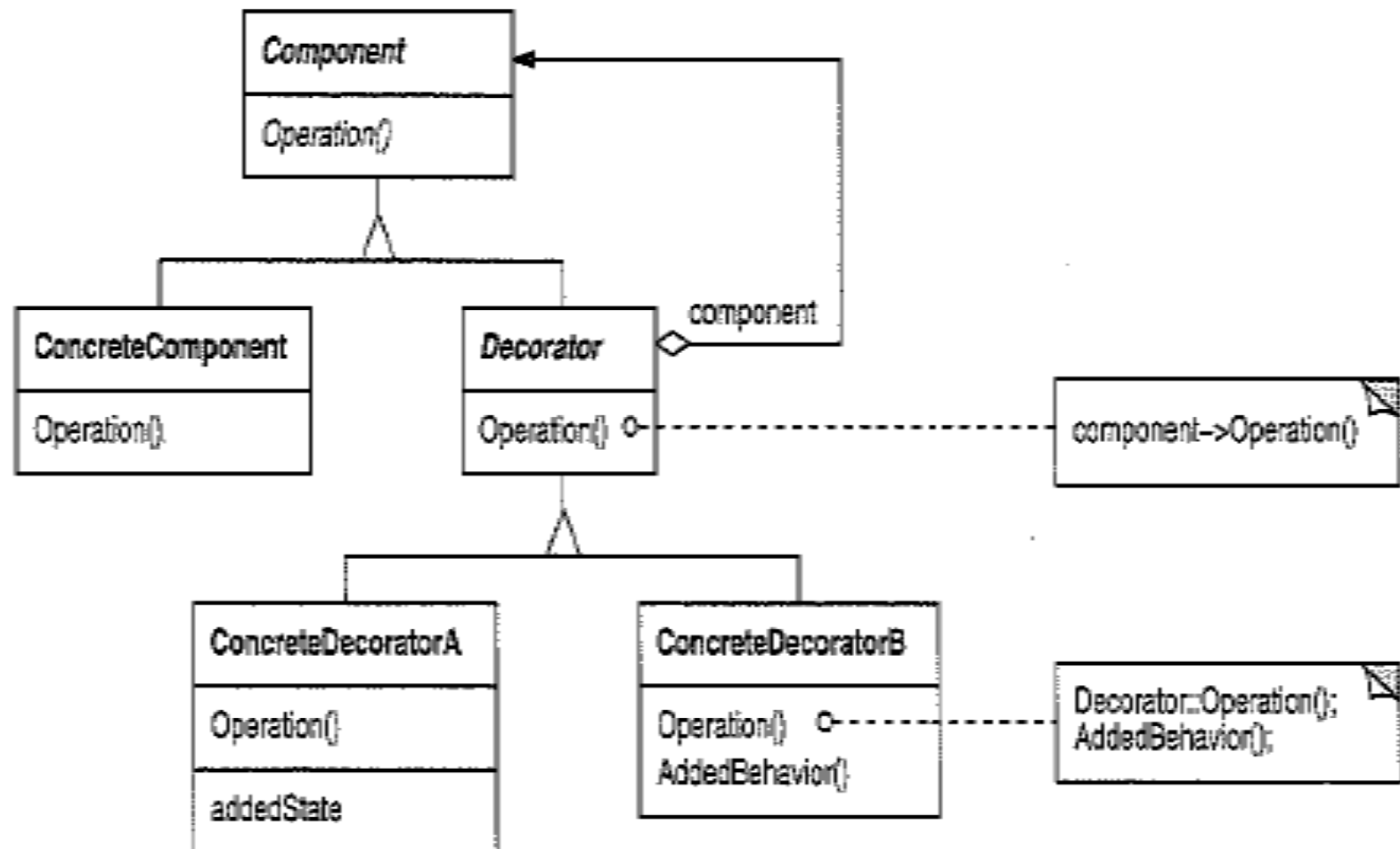
PDF viewer

- View document
  - Add scroll bar
- Add border

Car

- Base model
  - Add leather seats option
- Add sports package (upgraded wheels, shocks, etc)

# Decorator UML





# Sample instance structure

