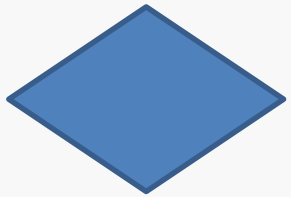
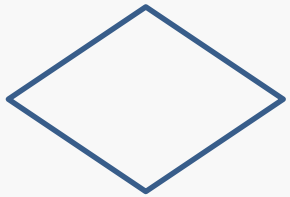
SWEN 383 Software Design Principles & Patterns

The Composite Pattern

But first…

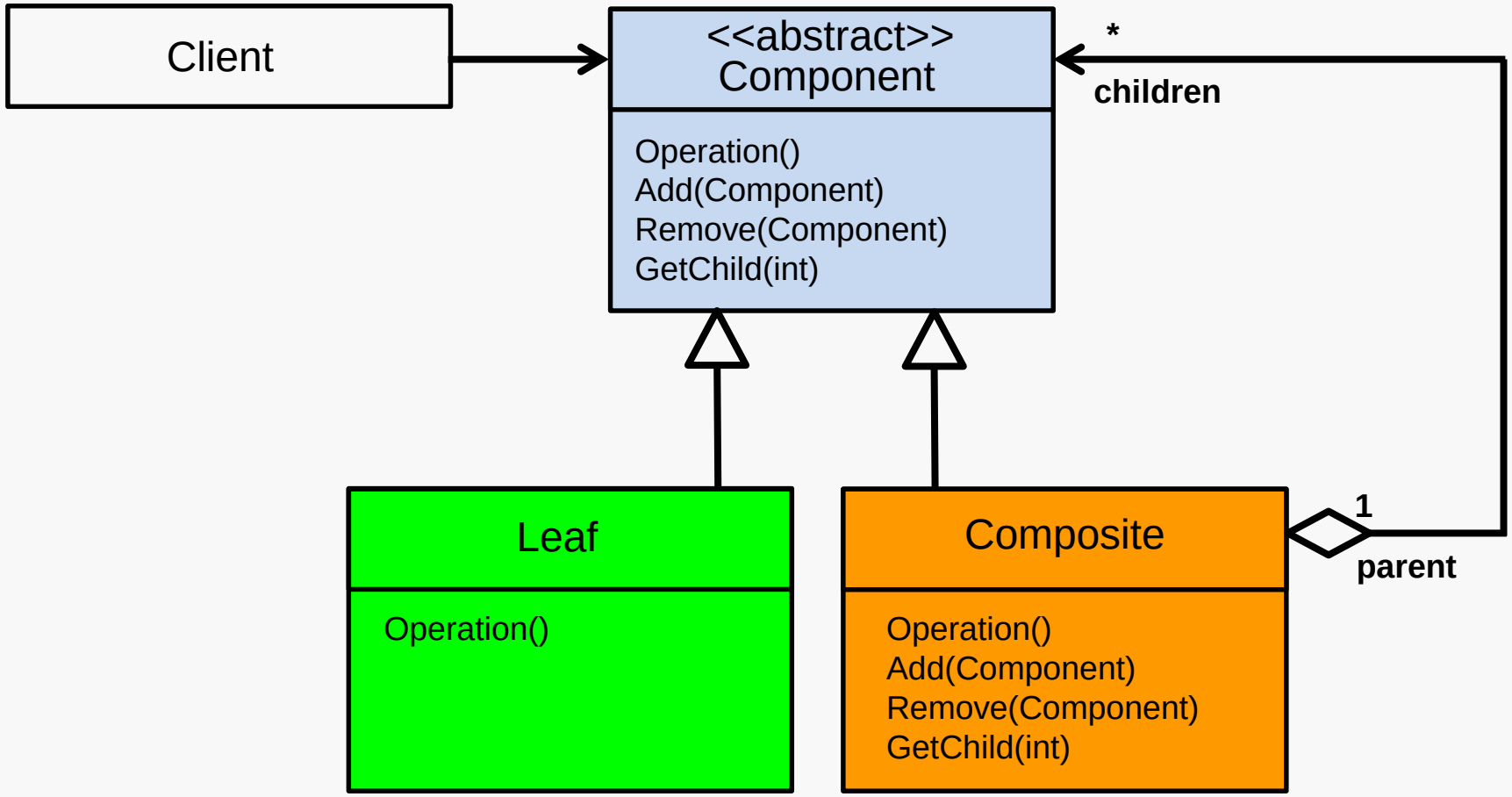


VS.



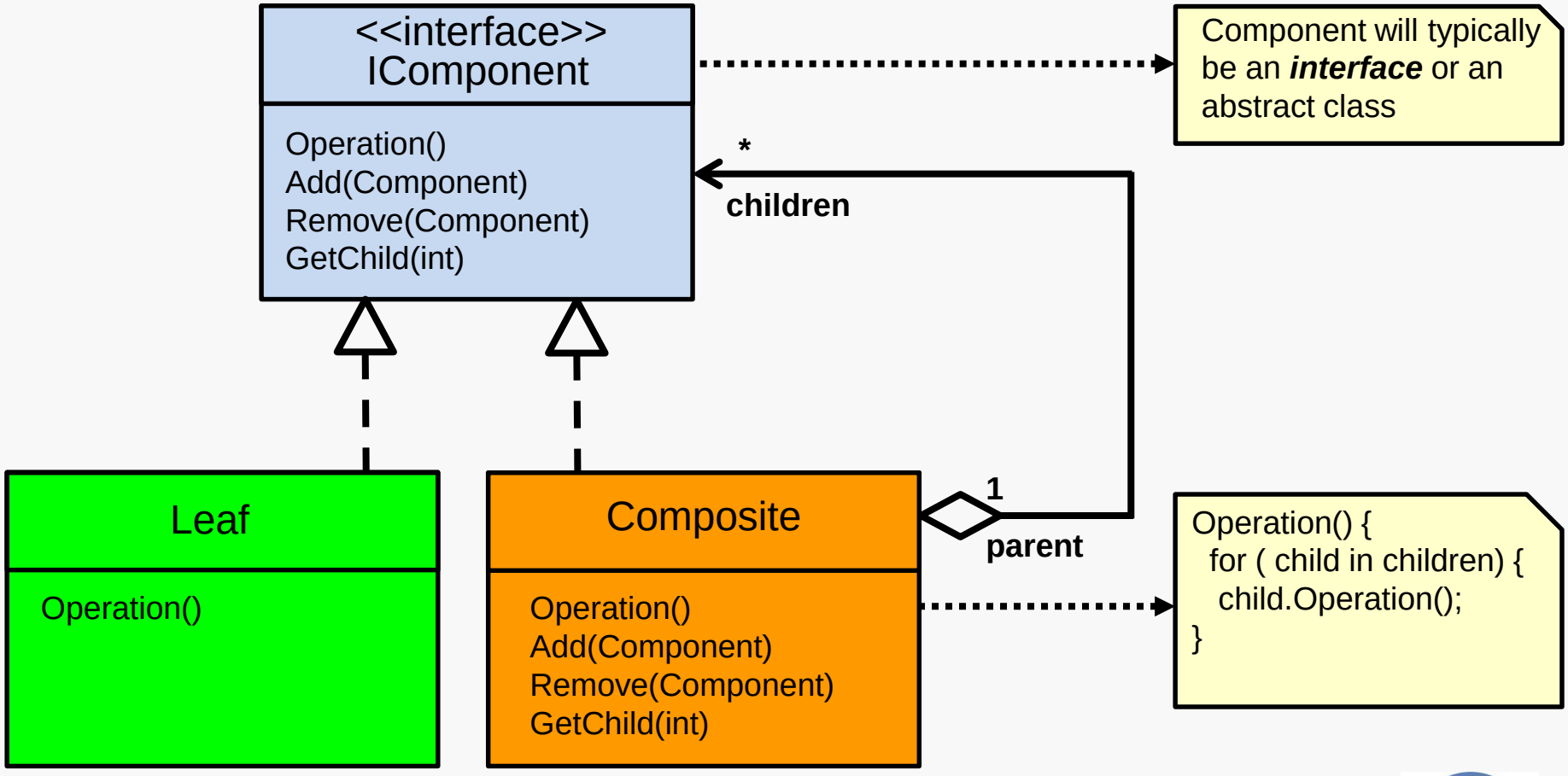
SWEN 383 Software Design Principles & Patterns

The Composite Pattern



SWEN 383 Software Design Principles & Patterns

The Composite Pattern

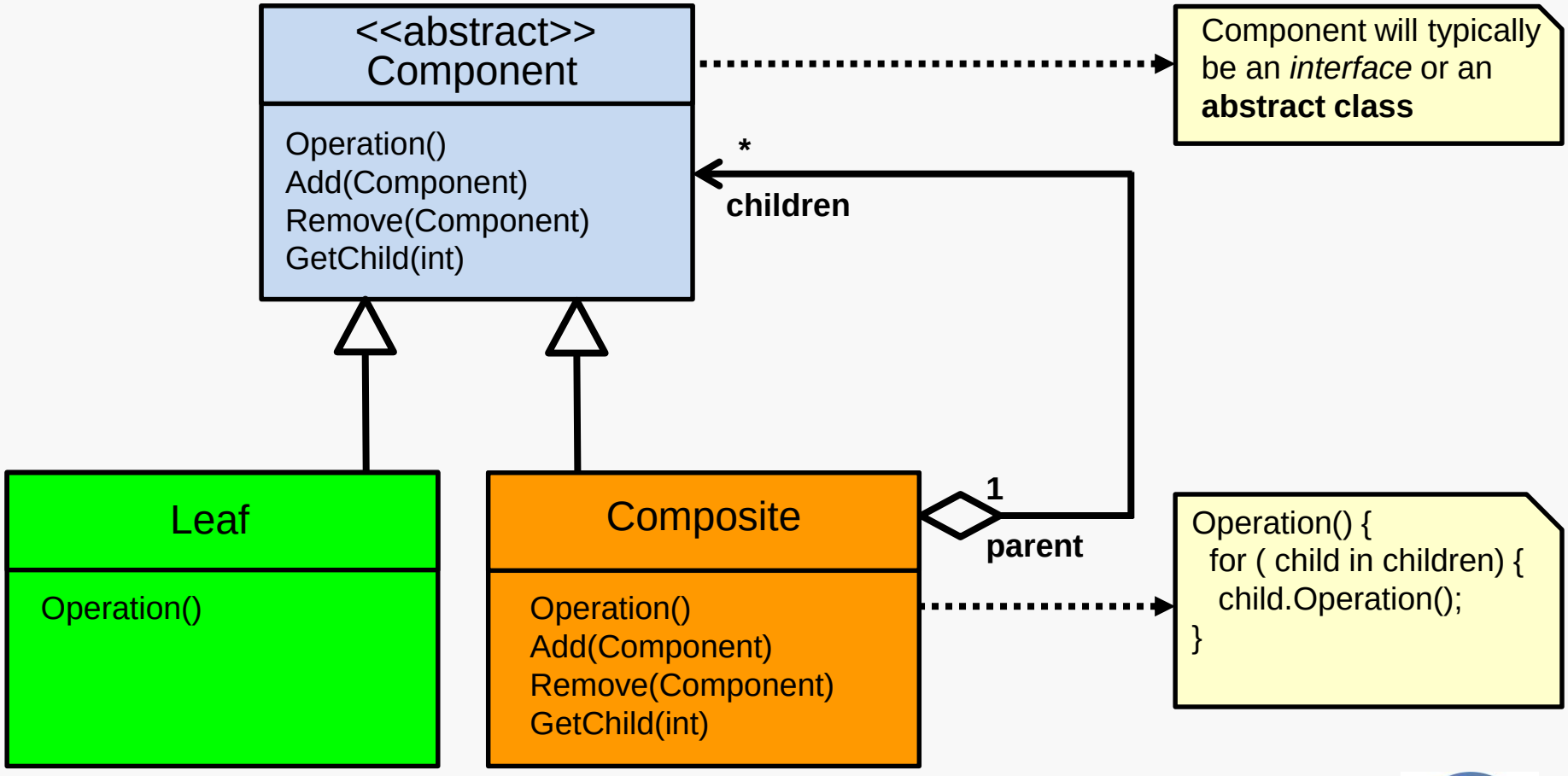


*

Software Engineering
Rochester Institute of Technology

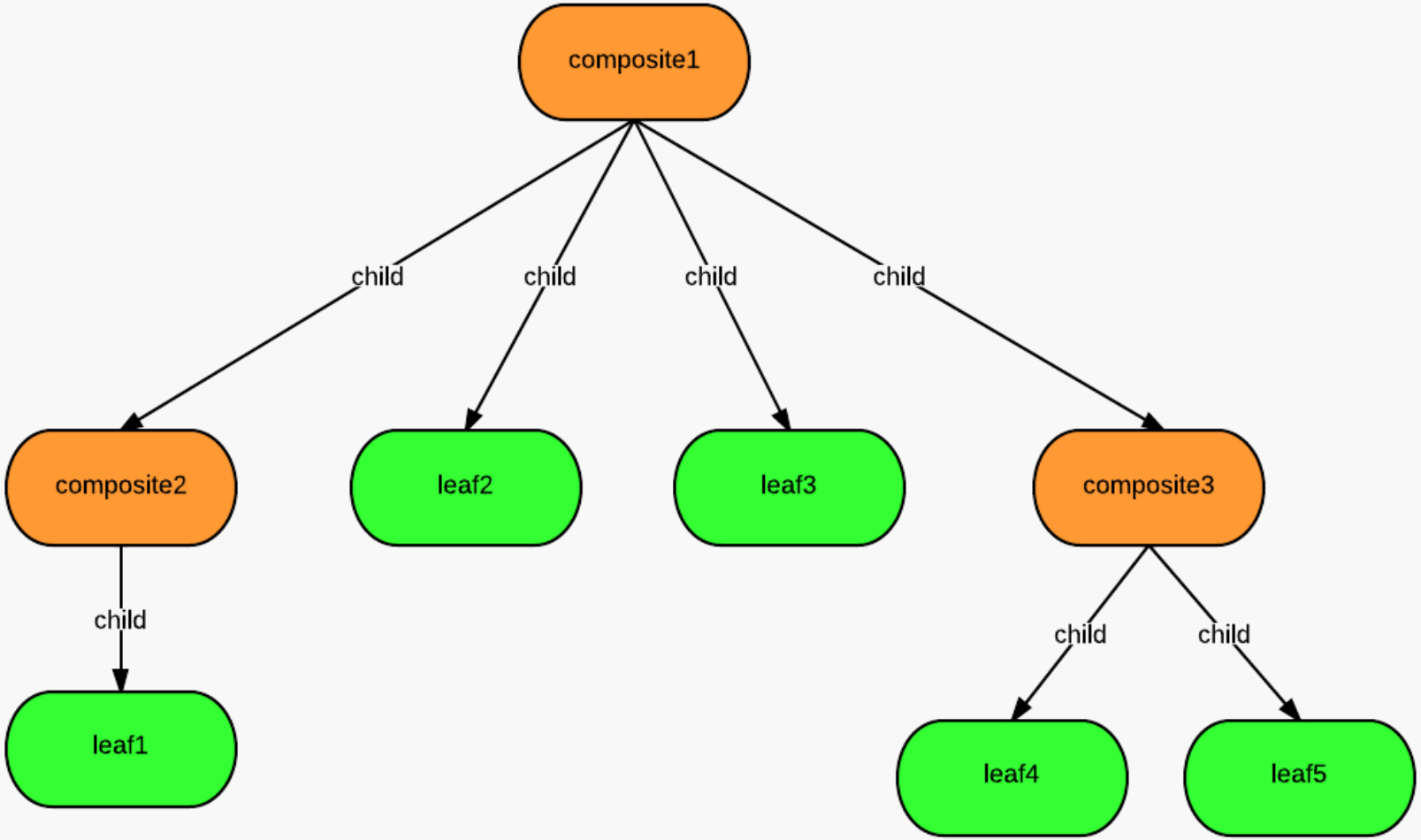
SWEN 383 Software Design Principles & Patterns

The Composite Pattern



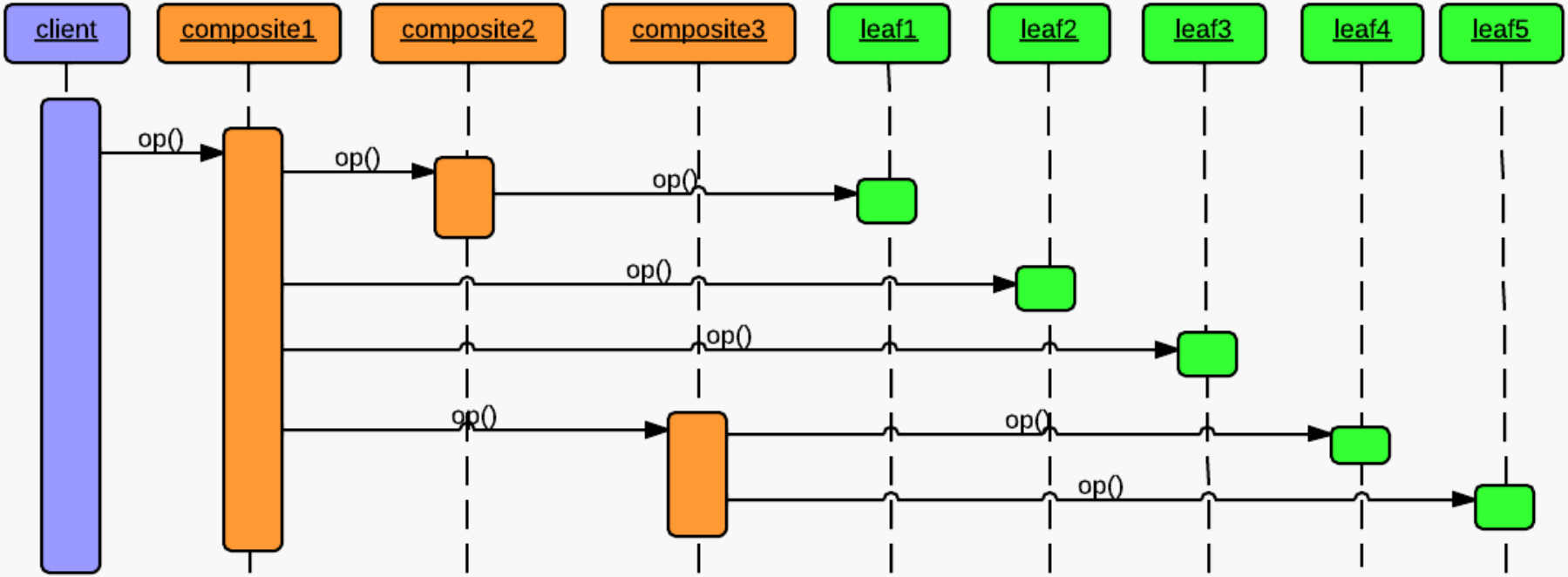
*

Composite Object Diagram



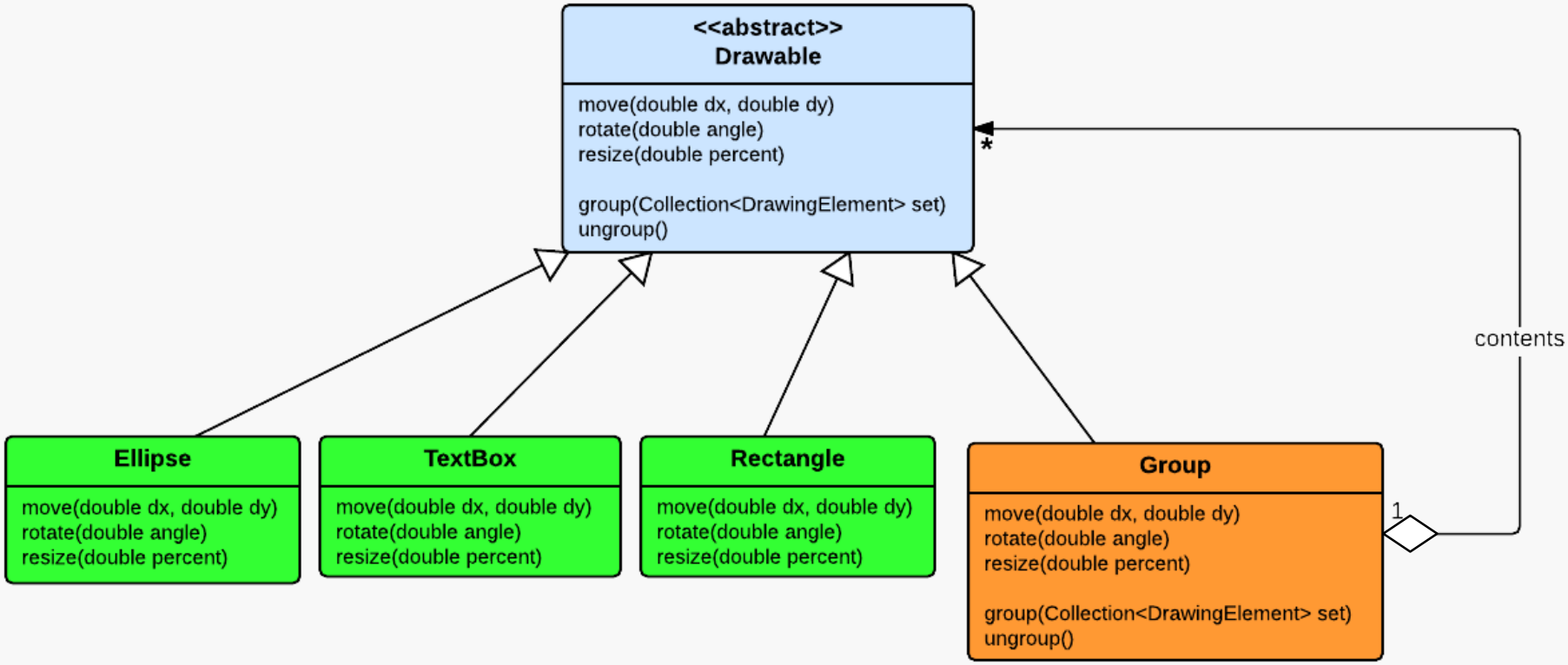
*

Composite Sequence Diagram



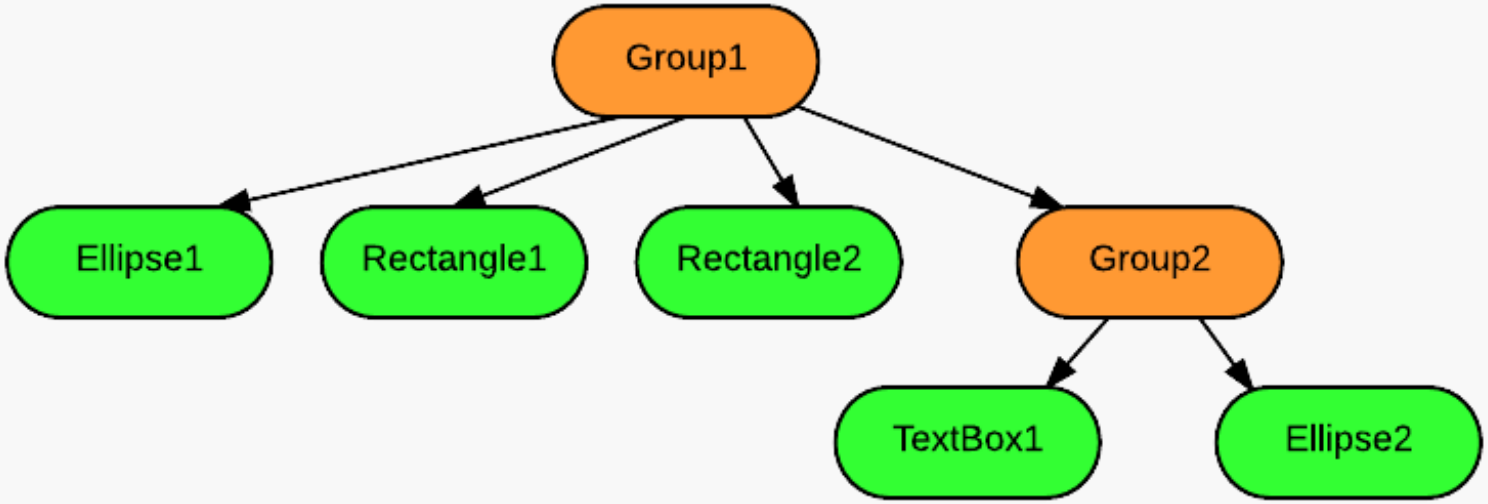
*

Composite Class Example



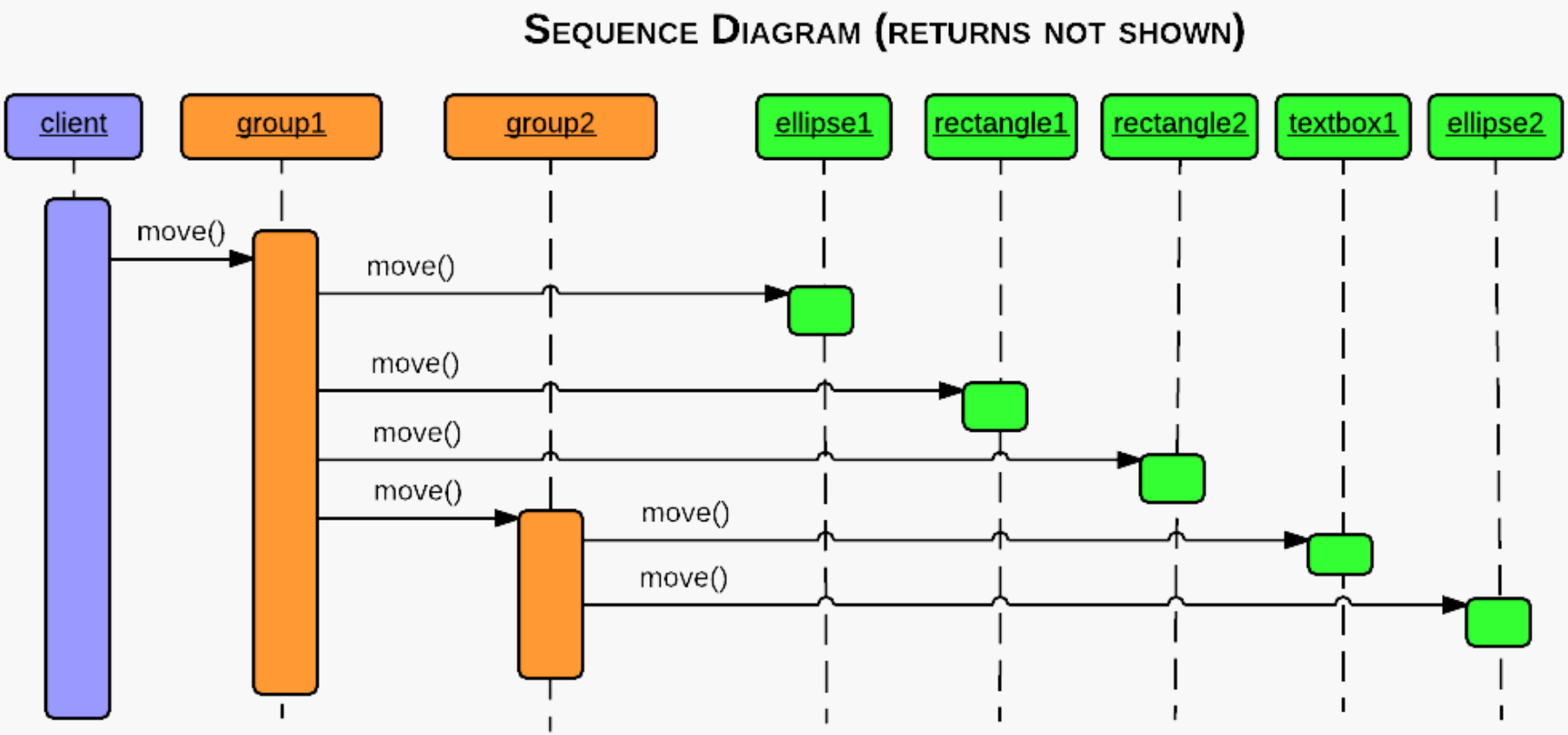
*

Composite Object Example



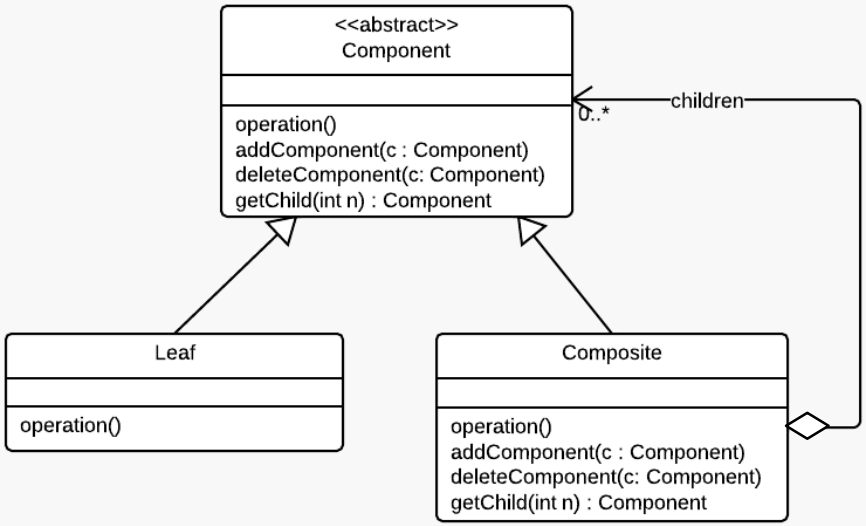
*

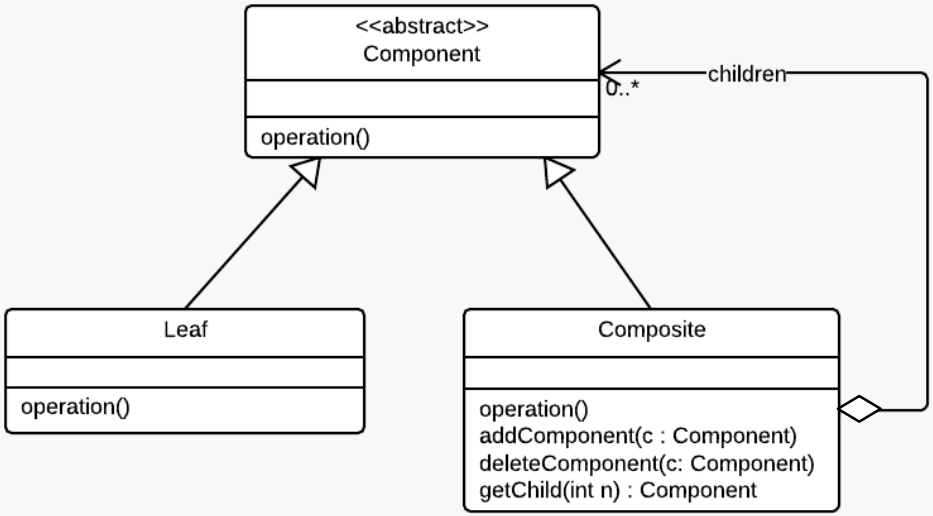
Composite Sequence Example



Discussion Questions (1 of 2)

Consider the two variants on the Composite pattern below:





What are the relative advantages and disadvantages of each approach?

Composite isComposite**() {…}**
- returns **null** for anything but Composites
- returns **this** for Composites

Software Engineering
Rochester Institute of Technology

Discussion Questions (2 of 2)

- Suppose we have several Composites.
  - What would be the advantages and disadvantages of allowing Composites to share children?
  - How might this complicate implementation?
- How might Composite be used create an internal object representation of an HTML page?
- In the Composite designs we've shown so far, navigation is unidirectional (from Composite to the Components it contains).
  - What would be an advantage and a disadvantage to having a reference to the containing Composite in each Component?
  - What about the case where Composites can share Components?