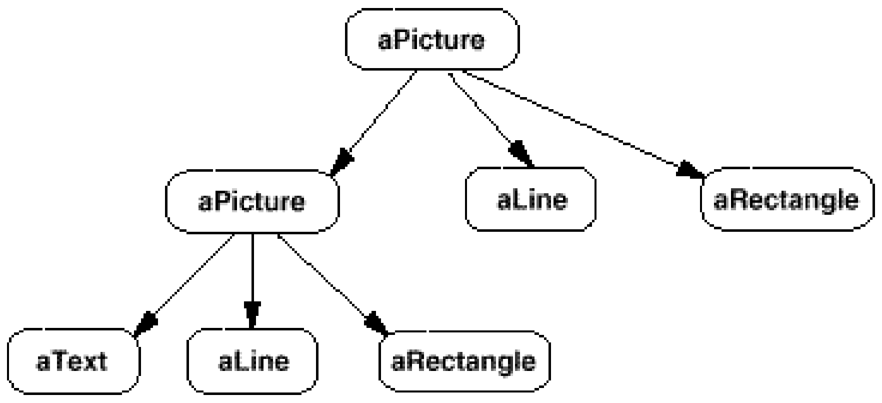
Composite Pattern

Alex X. Liu

Containment Relationship

- How to design the classes for line, rectangle, text, picture?
- A picture may contain some lines, and/or some rectangles, and/or some text, and/or some pictures.



Straightforward Design

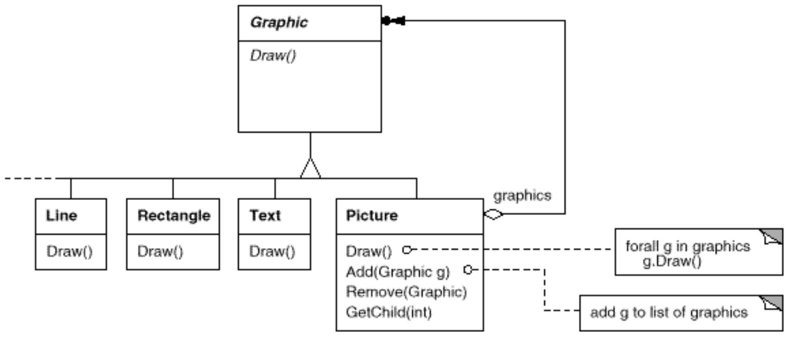
- Designing classes Line, Rectangle, Text is simple.
- How to design class Picture?
  - Need to have a vector of Line pointers, a vector of Rectangle pointers, a vector of Text pointers, and a vector of Picture pointers because a picture may contain zero of more lines, rectangles, text, or pictures.
  - For operations that needs to go through all the children of a picture, you need to have four loops over the four vectors respectively.
- This design is bad. Why?
  - 1. What if you want to add a Circle class? You have to change existing classes!
    - (1) Add a new vector of Circle pointers.
    - (2) For all operations in Picture class that need go through all children, you need to add another loop over the Circle vector.
- Principle: Anticipation of change.
  - Avoid making changes to existing classes.

Problem Characteristics

- **1. The objects have part-of relationship**, which are often recursive.
  - Two types of objects: leaves and composites
- **2. There are common operations.**
  - Often, to perform an operation for a composite object, we need to go over all its children.
- Example problems:
  - Graphics: line, rectangle, text, picture
    - Part-of relationship: a picture may contain lines, rectangles, texts, and pictures.
    - Common operations: draw()
  - Equipments: floppydisk, network card, chassis, cabinet
    - Part-of relationship : a chassis may contain floppydisks, network cards, etc; and a cabinet may contain floppydisks, network cards, chassis, and cabinets.
    - Common operations: power(), price()
  - Expressions: variables, literals, negate, binary expressions
    - Note: a binary expression is composed by two expressions (add, substract, multiply, divide).
    - Common operations: value(), print()

Composite Pattern

- Composite Pattern:
  - 1. Introduce an abstract base class that serves as the base for all leaf classes and composite classes. Define the common interfaces that all leaf classes and composite classes should have.
  - 2. All leaf classes and composite classes inherit from this abstract class.
  - 3. In each composite class, include a vector of abstract class pointers.
- Design virtues
  - 1. Avoid making changes to existing classes: add new leaf classes and composite classes without changing existing composite classes.
  - 2. Polymorphism


Graphic
Draw()
Line
Draw()
Rectangle
Draw()
Text
Draw()
Picture
Draw()
Add(Graphic g)
Remove(Graphic)
GetChild(int)
graphics
forall g in graphics
g.Draw()
add g to list of graphics