Strategy Pattern

Shin-Jie Lee (李信杰)
Associate Professor
Computer and Network Center
Department of CSIE
National Cheng Kung University

Design Aspect of Strategy

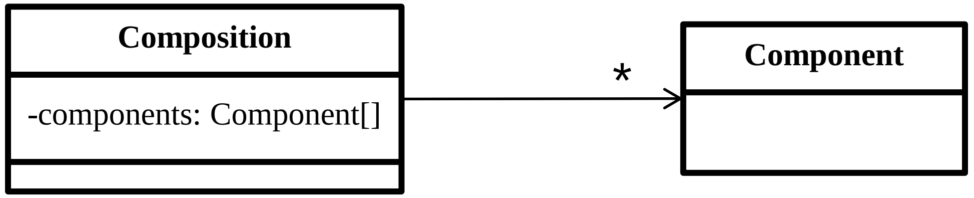
An algorithm

Outline

Text Composition Design (Strategy)

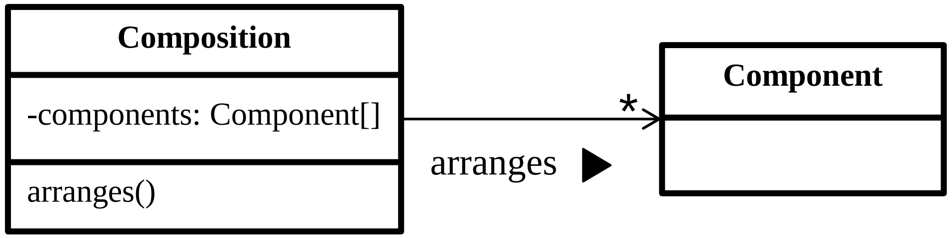
Requirements Statement$_1$

- The Composition class maintains a collection of Component instances, which represent text and graphical elements in a document.



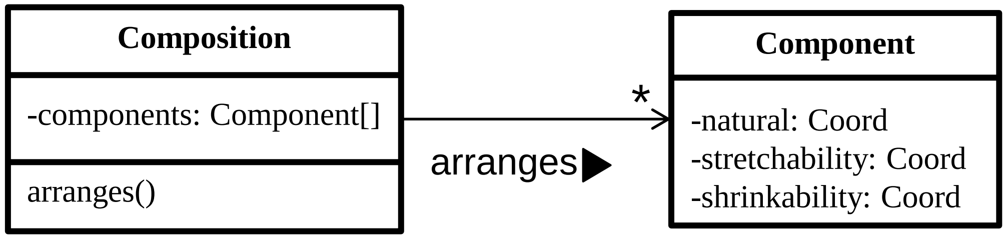
Requirements Statement$_2$

- A composition arranges component objects into lines using a linebreaking strategy.



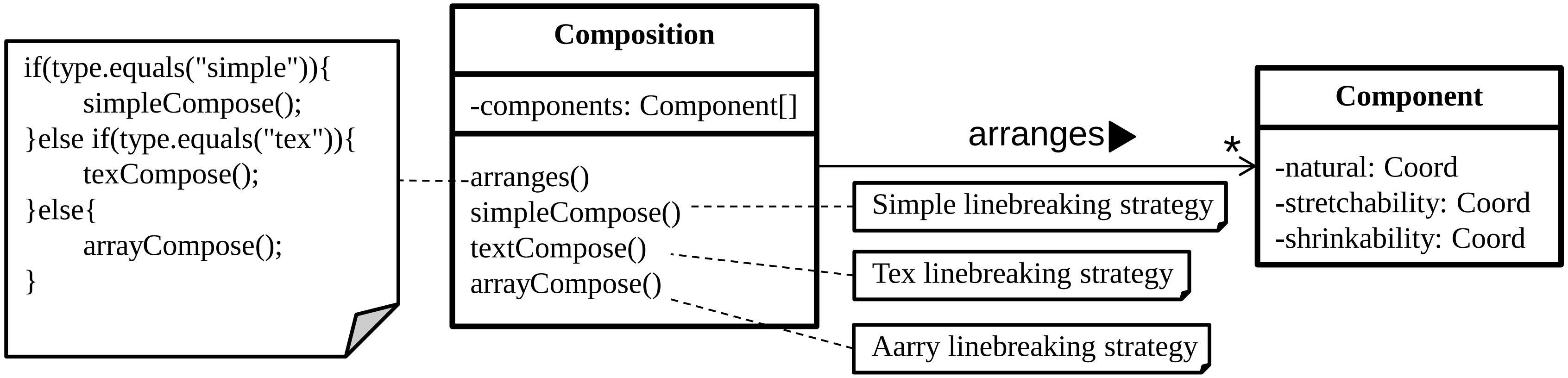
Requirements Statement$_3$

- Each component has an associated natural size, stretchability, and shrinkability.
- The stretchability defines how much the component can grow beyond its natural size; shrinkability is how much it can shrink.



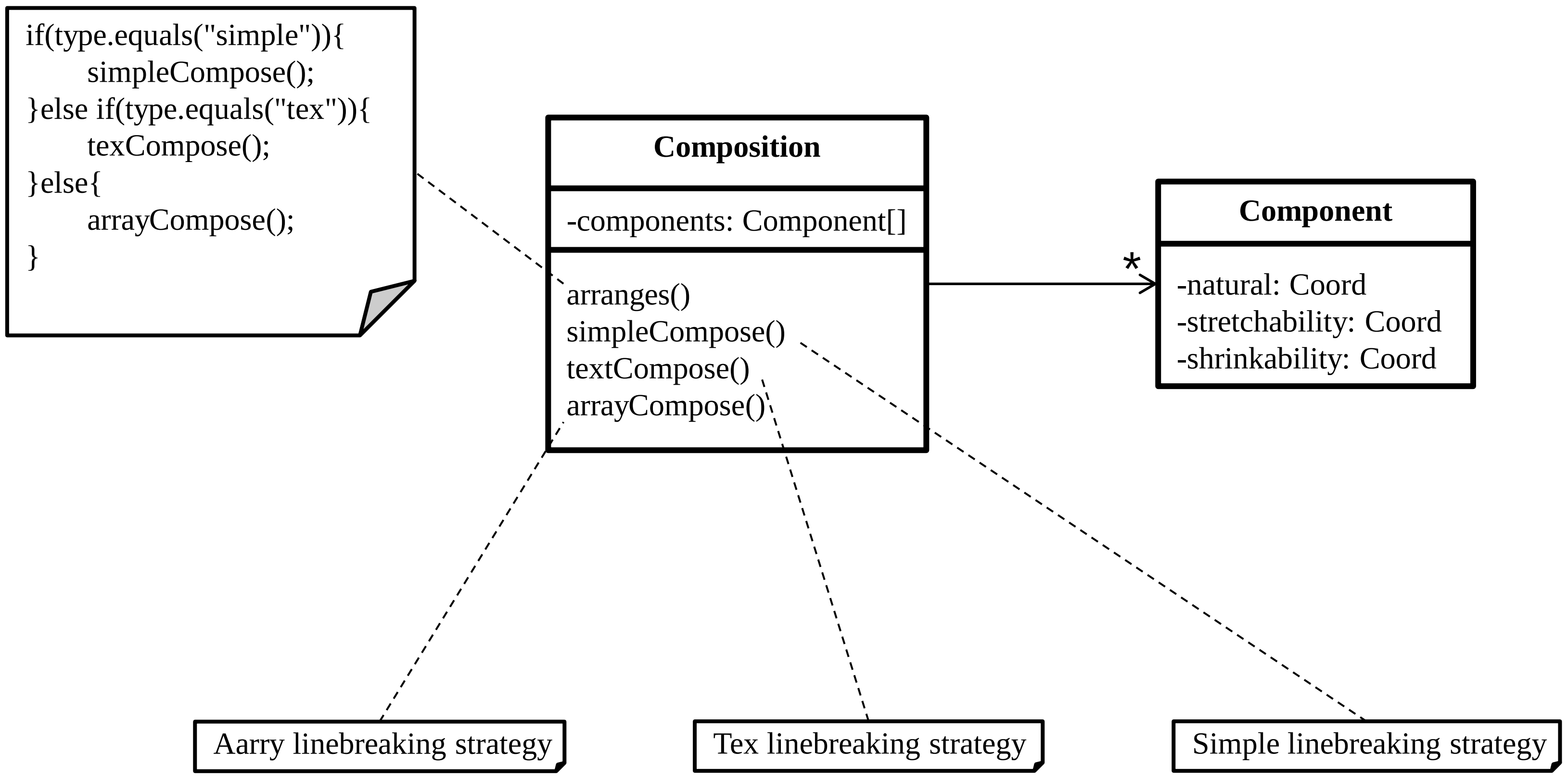
Requirements Statement$_4$

- ❑ When a new layout is required, the composition calls its compose method to determine where to place linebreaks.
- ❑ There are 3 different algorithms for breaking lines:
  - ➢ **Simple Composition:** A simple strategy that determines line breaks one at a time.
  - ➢ **Tex Composition:** This strategy tries to optimize line breaks globally, that is, one paragraph at a time.
  - ➢ **Array Composition:** A strategy that selects breaks so that each row has a fixed number of items. It's useful for breaking a collection of icons into rows, for example.

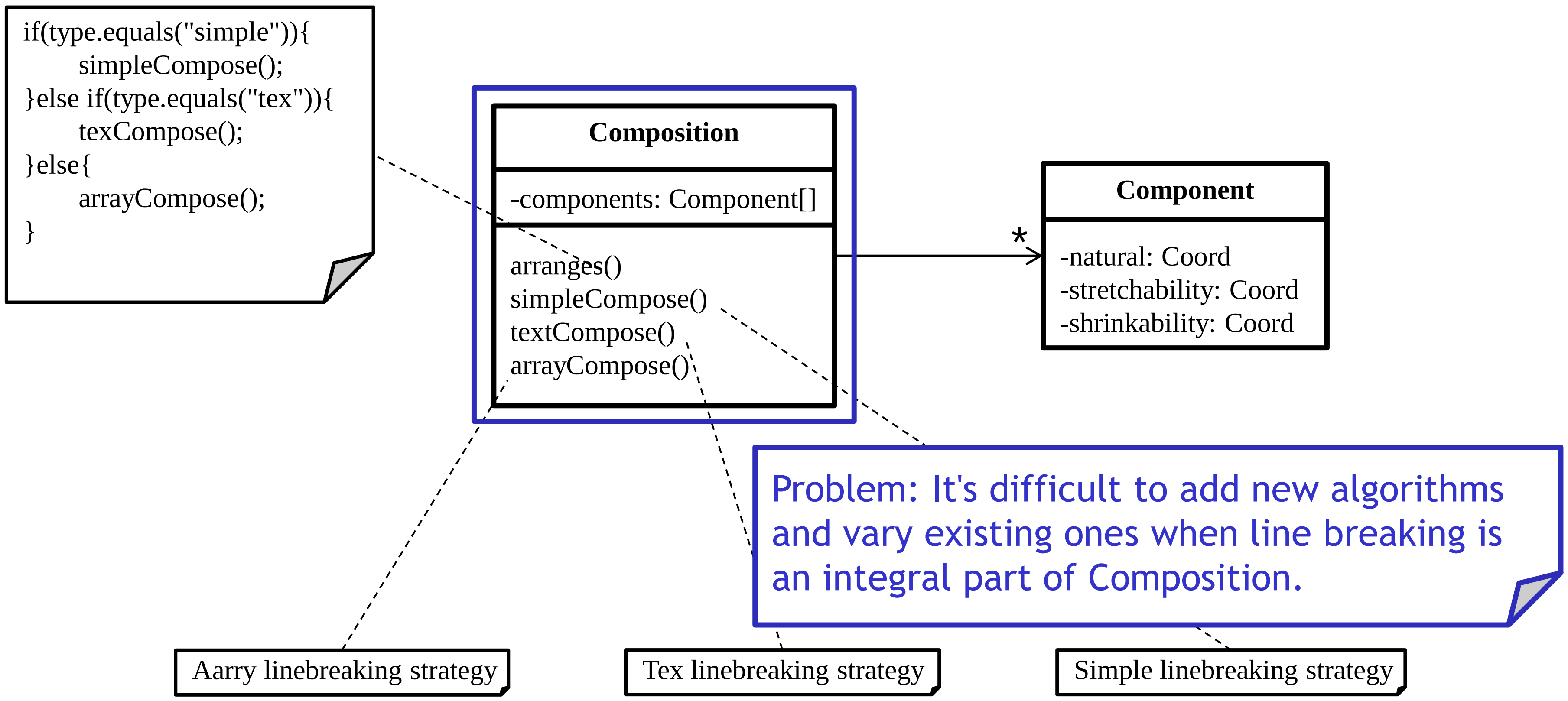


```
if(type.equals("simple")){
        simpleCompose();
}else if(type.equals("tex")){
        texCompose();
}else{
        arrayCompose();
}
```

Initial Design


```
if(type.equals("simple")){
        simpleCompose();
}else if(type.equals("tex")){
        texCompose();
}else{
        arrayCompose();
}
```

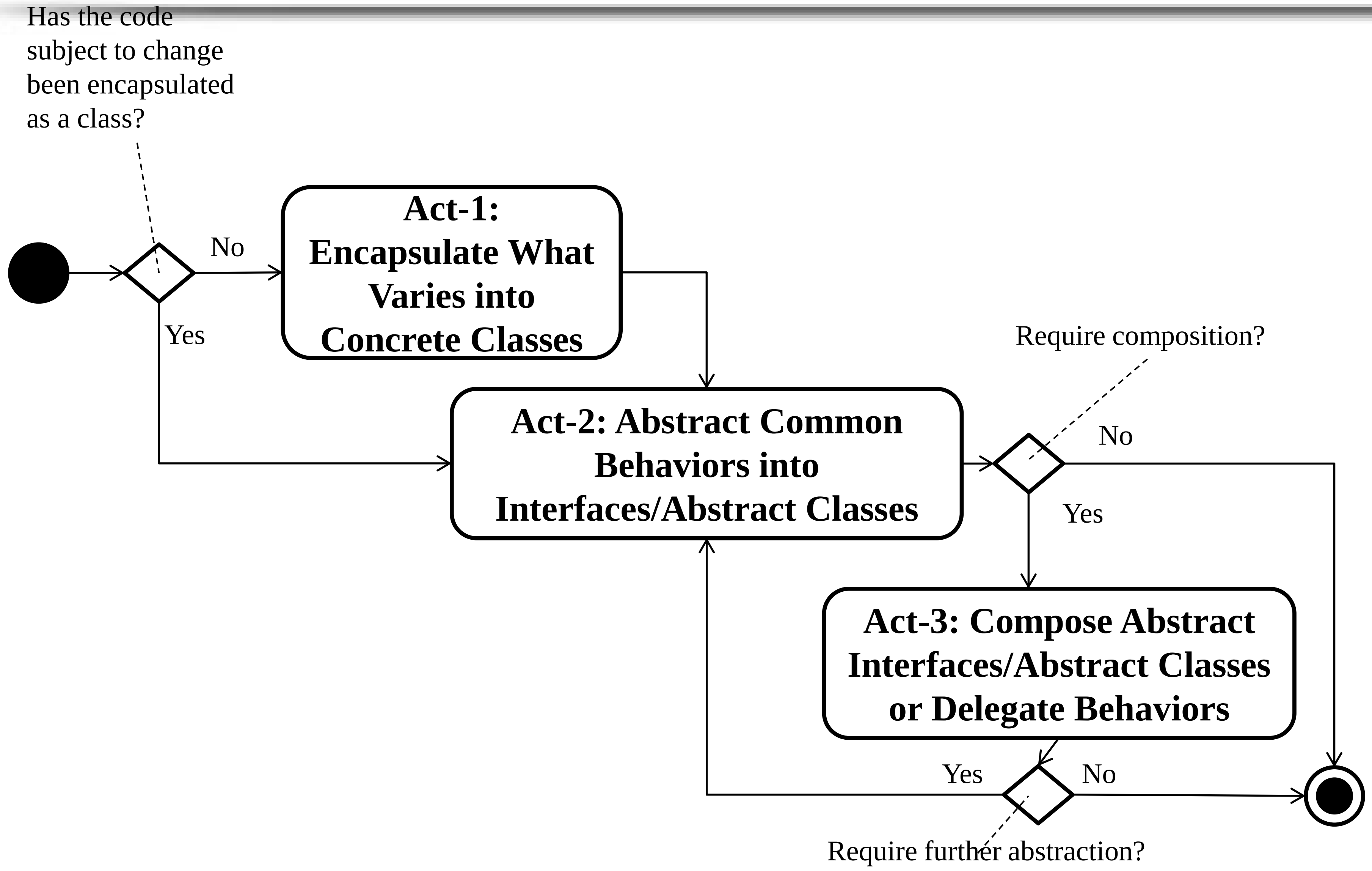
Problems with Initial Design



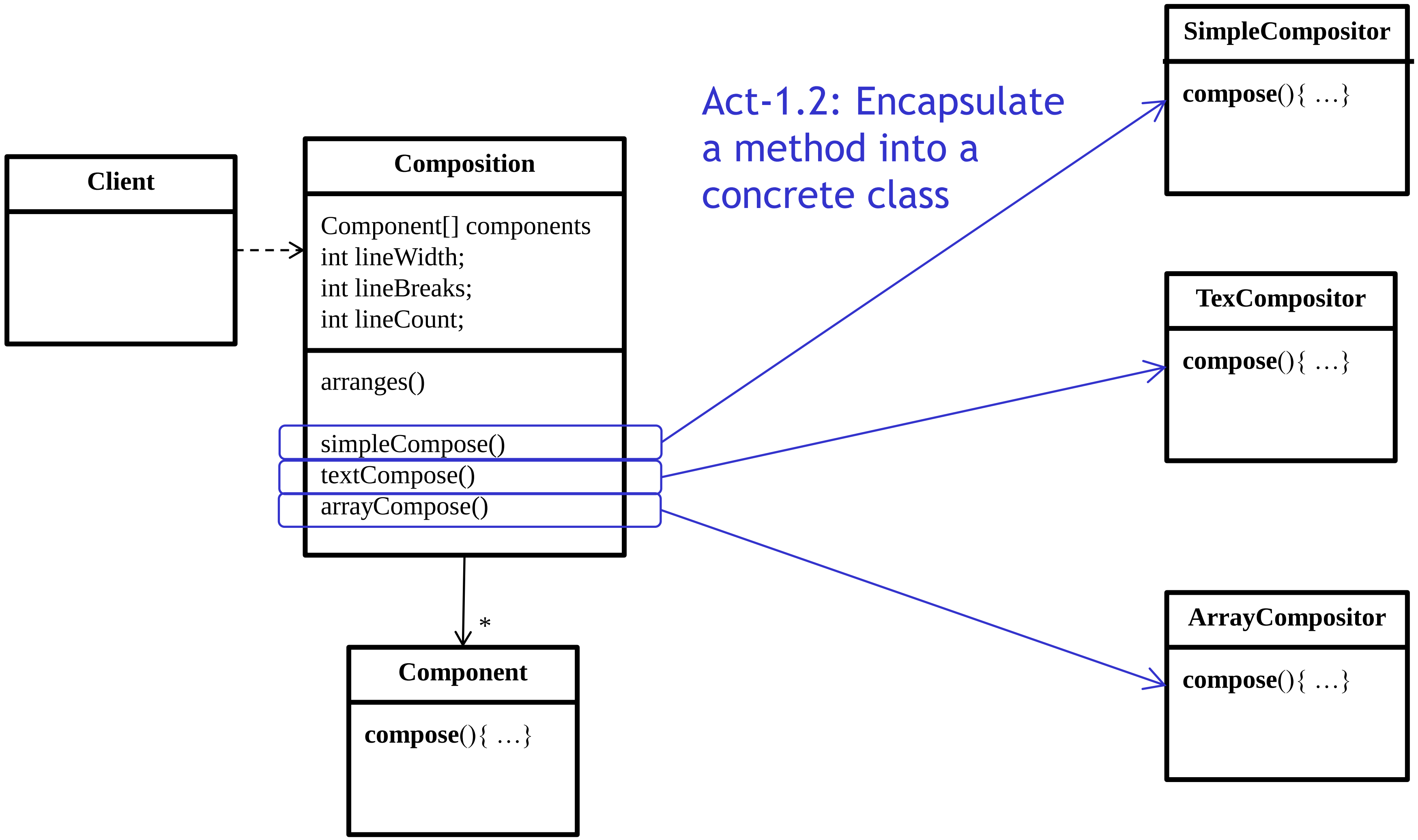
```
if(type.equals("simple")){
        simpleCompose();
}else if(type.equals("tex")){
        texCompose();
}else{
        arrayCompose();
}
```

Problem: It's difficult to add new algorithms and vary existing ones when line breaking is an integral part of Composition.

Design Process for Change

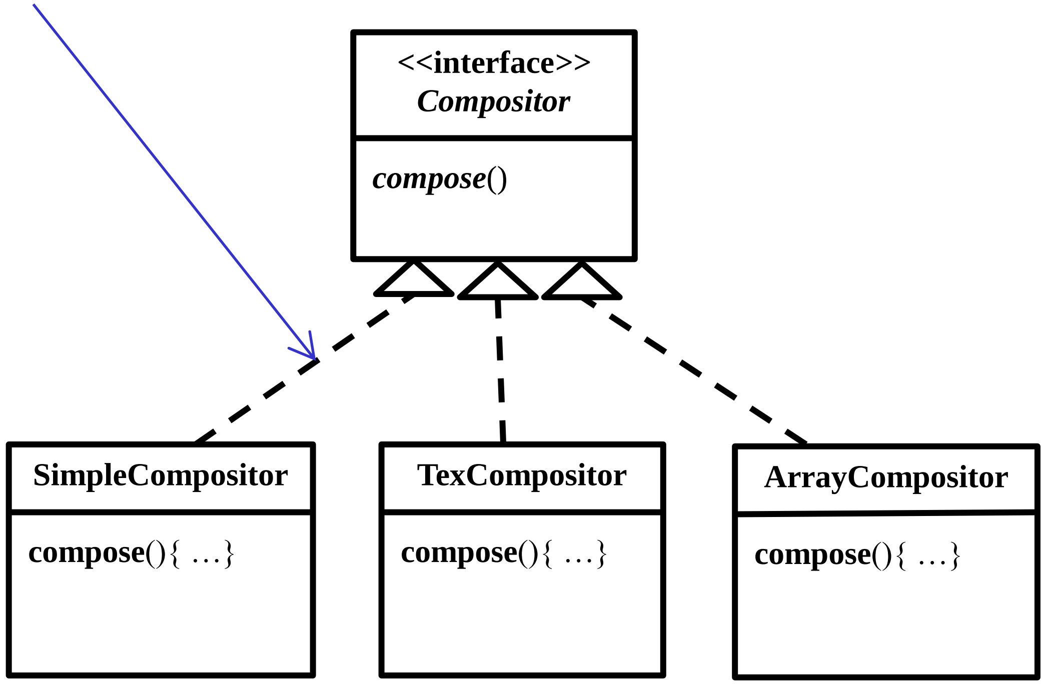

Has the code subject to change been encapsulated as a class?
No
Yes
Act-1: Encapsulate What Varies into Concrete Classes
Act-2: Abstract Common Behaviors into Interfaces/Abstract Classes
Require composition?
No
Yes
Act-3: Compose Abstract Interfaces/Abstract Classes or Delegate Behaviors
Yes
No
Require further abstraction?

Act-1: Encapsulate What Varies

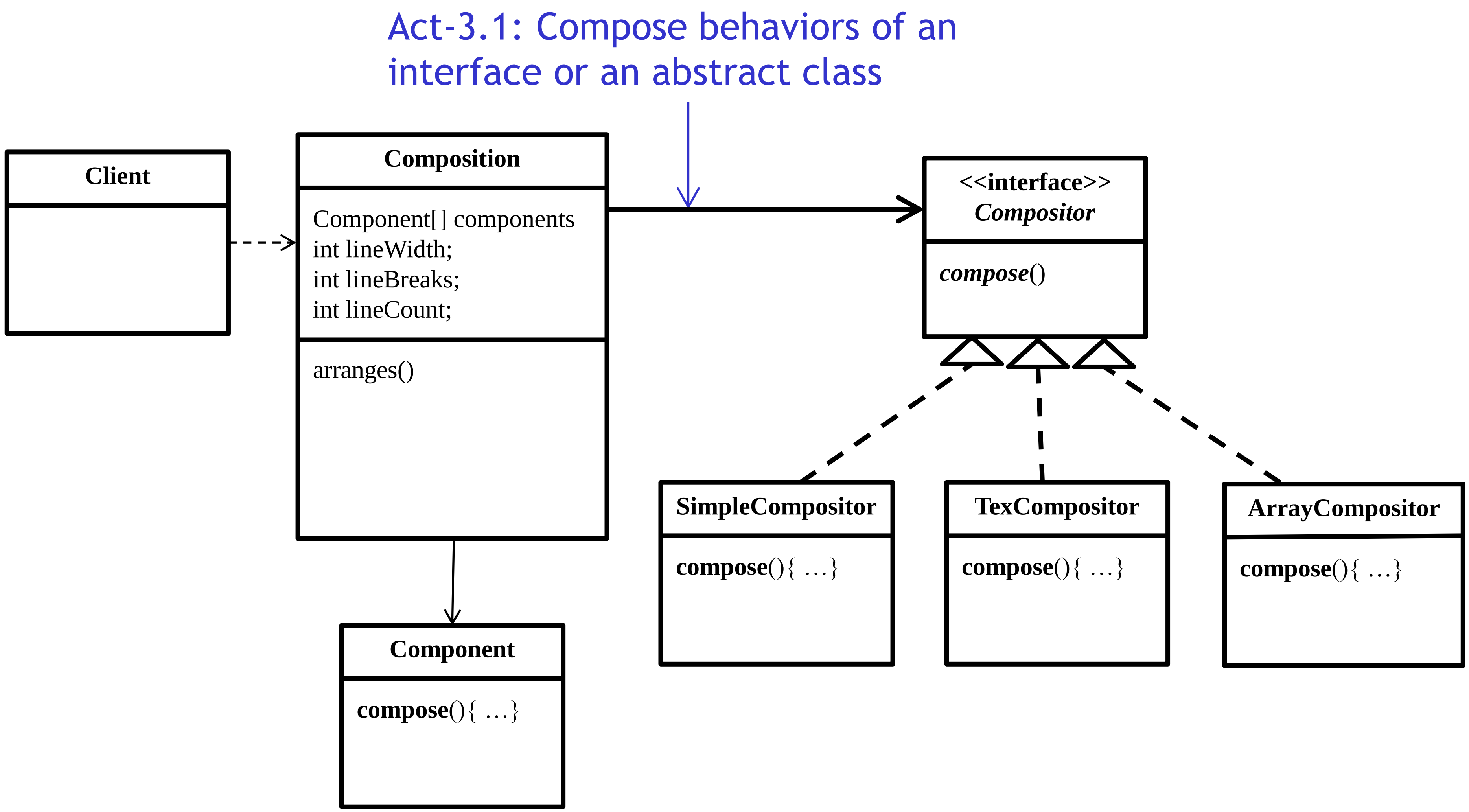


Act-2: Abstract Common Behaviors

Act-2.1: Abstract common behaviors with a same signature into interface through polymorphism



Act-3: Compose Abstract Behaviors



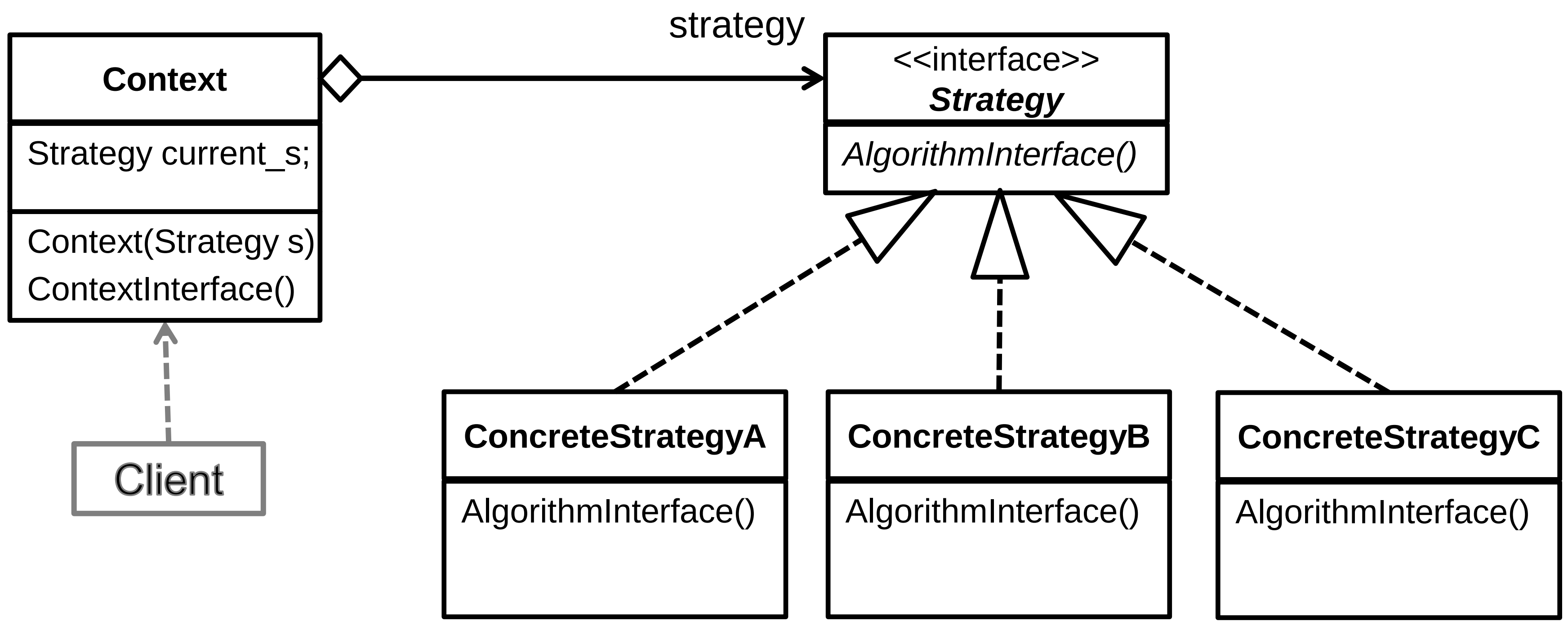
Recurrent Problems

- Multiple classes will be modified if new behaviors are to be added.
  - It's difficult to add new algorithms and vary existing ones.
- All duplicate code will be modified if the behavior is to be changed.
  - Different algorithms will be appropriate at different times.

Intent

- Define a family of algorithms, encapsulate each one, and make them interchangeable.
- Strategy lets the algorithm vary independently from clients that use it.

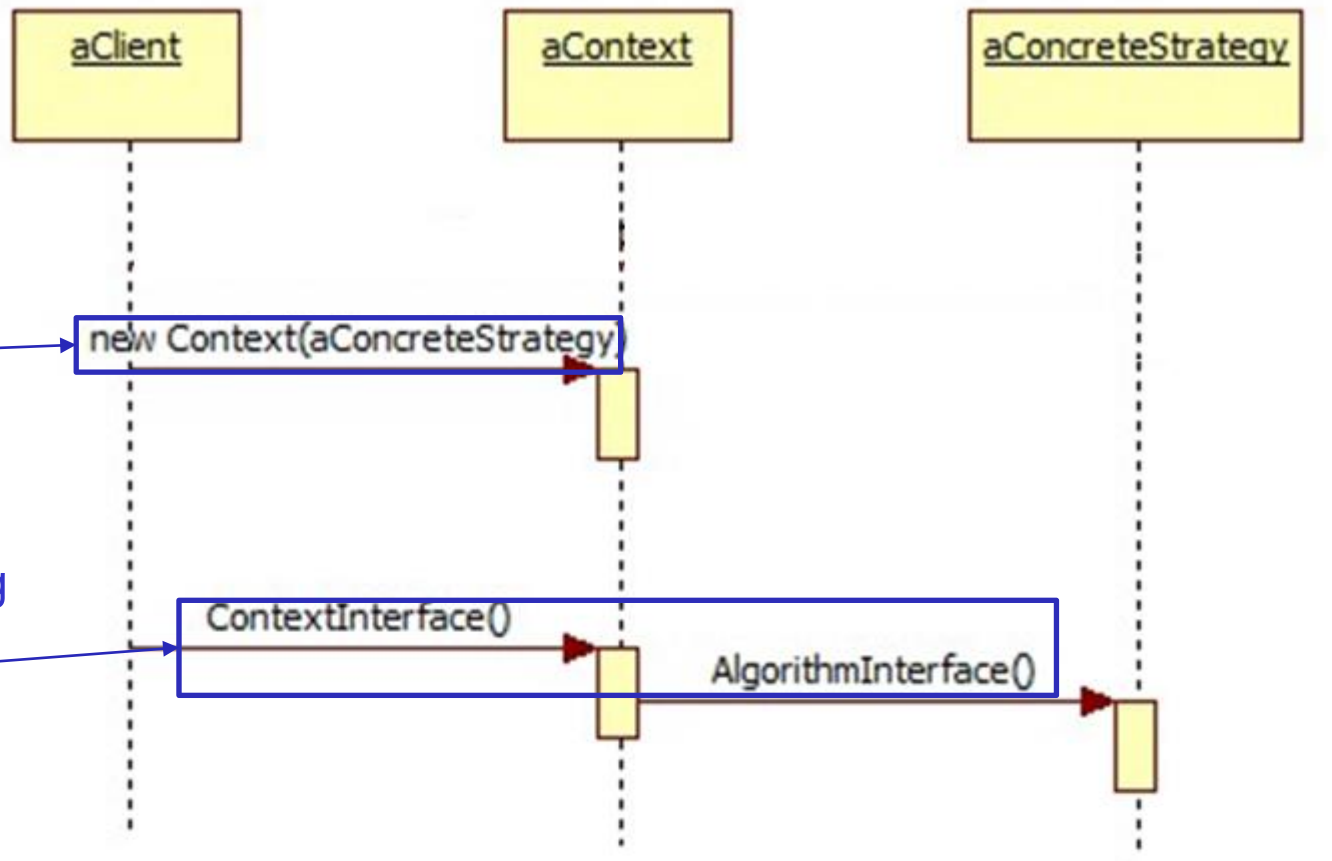
Strategy Pattern Structure1



Strategy Pattern Structure2

1. Client assigns a concrete strategy to Context.

2. Client perform the strategy through invoking Context's method.



Strategy Pattern Structure3

	Instantiation	Use	Termination
Client	Other class except classes in the strategy pattern	Other class except classes in the strategy patterns	Other class except classes in the strategy pattern
Context	Other class or the client class	Client passes a ConcreteStrategy reference to this class and delegates request to it	Other class or the client class
Strategy	X	Context uses this interface to call the algorithm defined by a ConcreteStrategy	X
Concrete Strategy	The client class or other class except classes in the strategy pattern	Context uses this class which is passed through reference by client through polymorphism	Classes who hold the reference of ConcreteStrategy

Lab: Ducks Game

Requirements Statement

- There are four types of ducks in the game: MallardDuck, RedheadDuck, RubberDuck, and DecoyDuck.
- All types of the ducks have the same swim behavior but are with different displays.
- Some ducks can fly with wings, but some cannot fly.
- A duck can quack, squeak, or be silent.