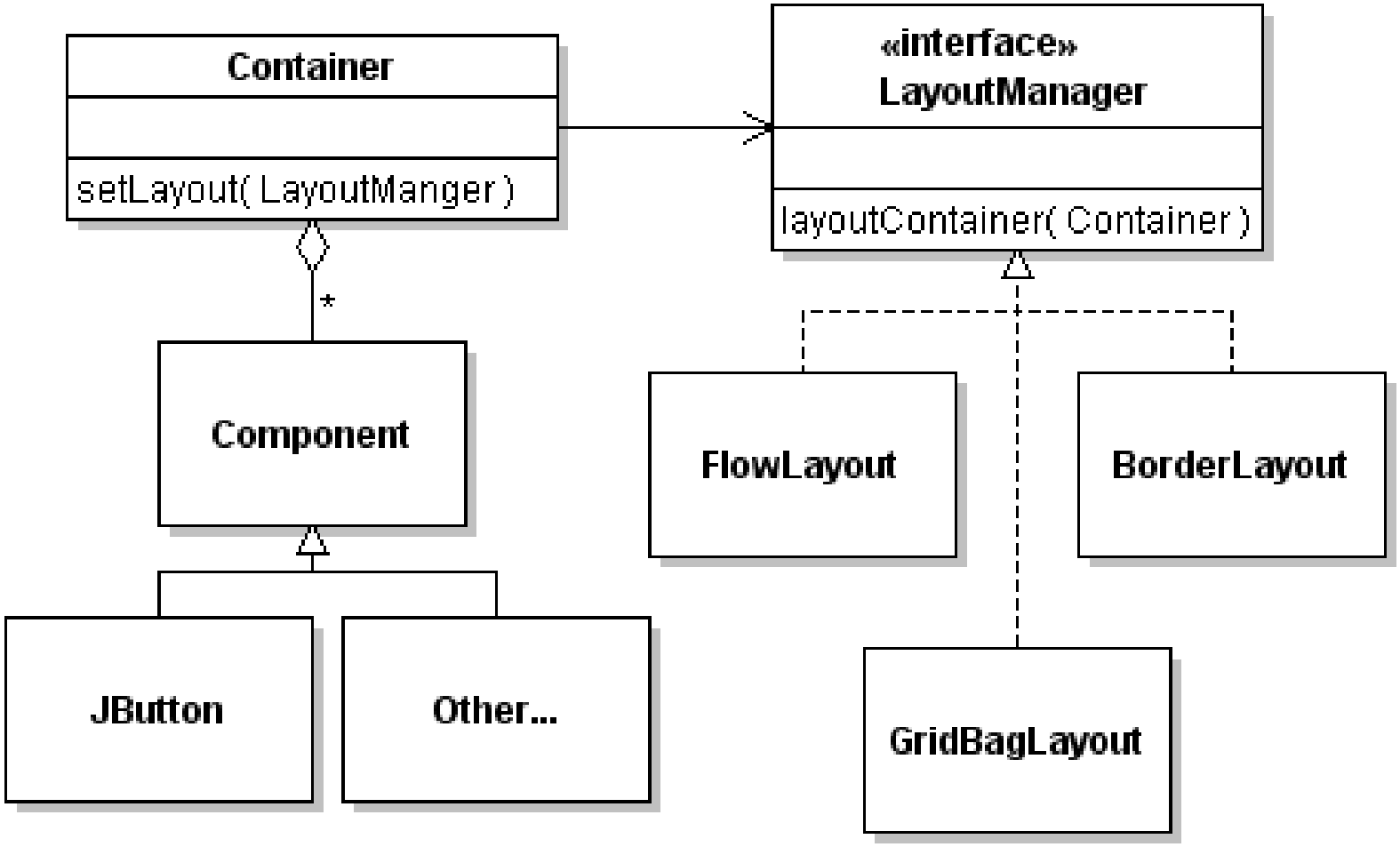
Design Patterns in Swing and AWT

- Strategy Pattern: LayoutManagers are *Strategies*
- Observer Pattern: event listeners are observers
- Composite Pattern: a container can be placed inside another container just like any component
- Decorator Pattern: you can "decorate" any component by using JScrollPane to add scroll bars
- MVC Pattern: JTable gets data from a TableModel
- Command Pattern: Action objects are commands that are invoked by components (like JButton). The Invoker, the Command, and Receiver (your application logic) are all separate.

LayoutManager - what pattern?



LayoutManager is a Strategy

Strategy Pattern

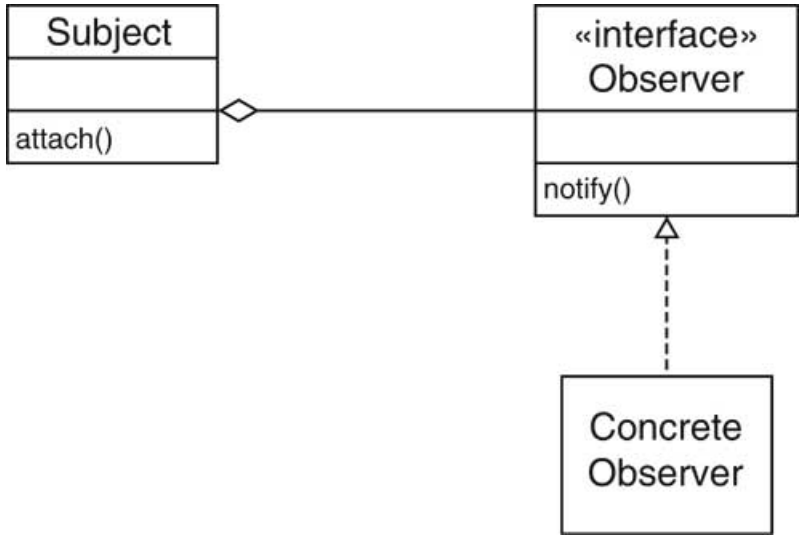
Name in Pattern	Name in this Example
Content	Container
Strategy	LayoutManager
Concrete Strategy	FlowLayout, GridBagLayout, ..
setStrategy	setLayout(LayoutManager)
doWork( )	layoutContainer( )

Benefit of LayoutManager

- What are the benefits of separating LayoutManager from the container classes?

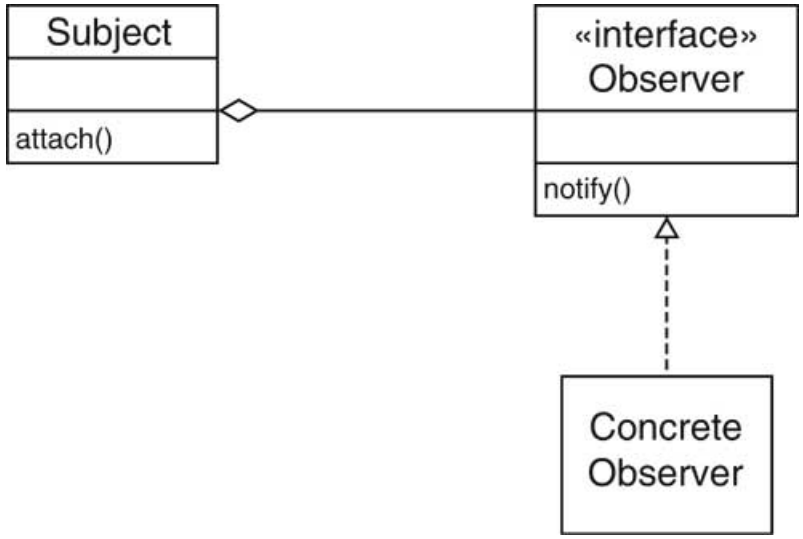
  Why don't we put the layout code inside each container?

Observer Pattern



Name In Observer Pattern	Name in Swing graphics
Subject	
Observer	
Concrete Observer	
attach( )	
notify( )	

Observer Pattern



Name In Observer Pattern	Name in Swing graphics
Subject	JButton, JMenuItem, JCheckBox
Observer	ActionListener
Concrete Observer	your class implementing ActionListener
attach( )	`addActionListener( observer )`
notify( )	`actionPerformed( )`