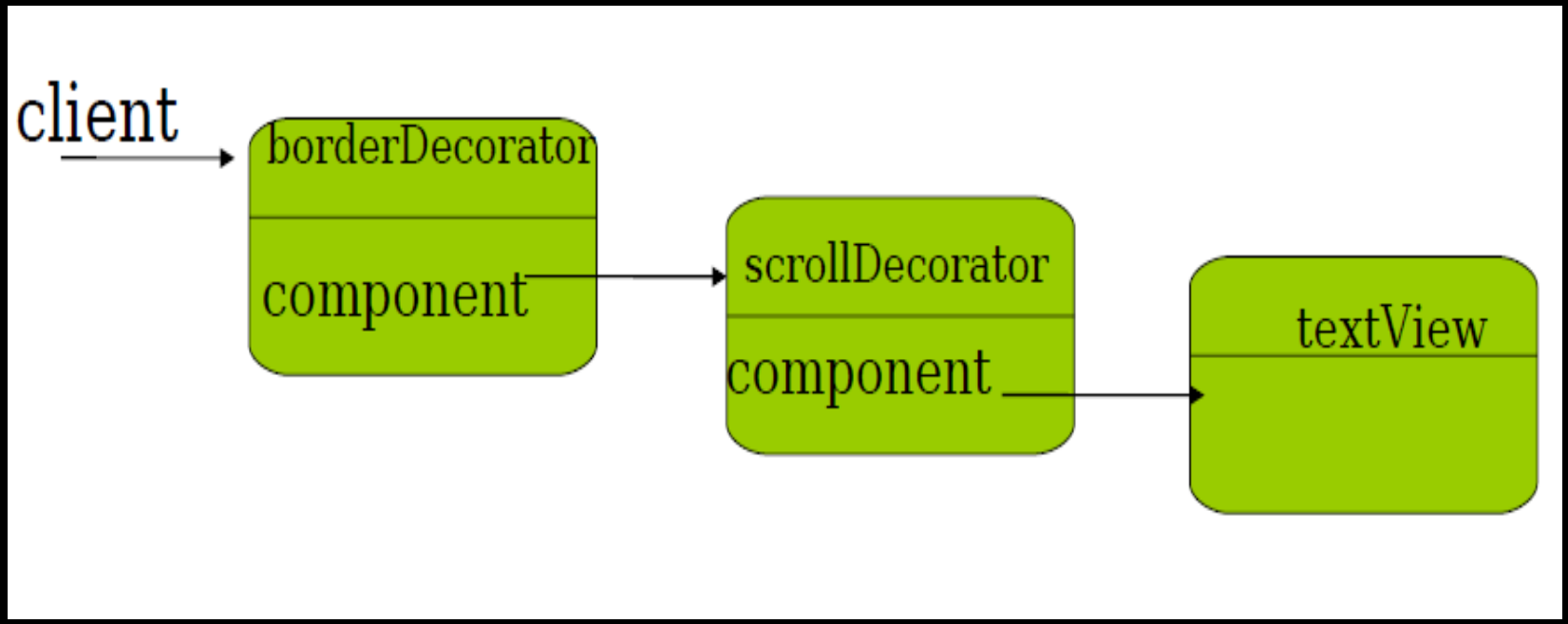
Reuse at Design Level: Design Patterns – IV

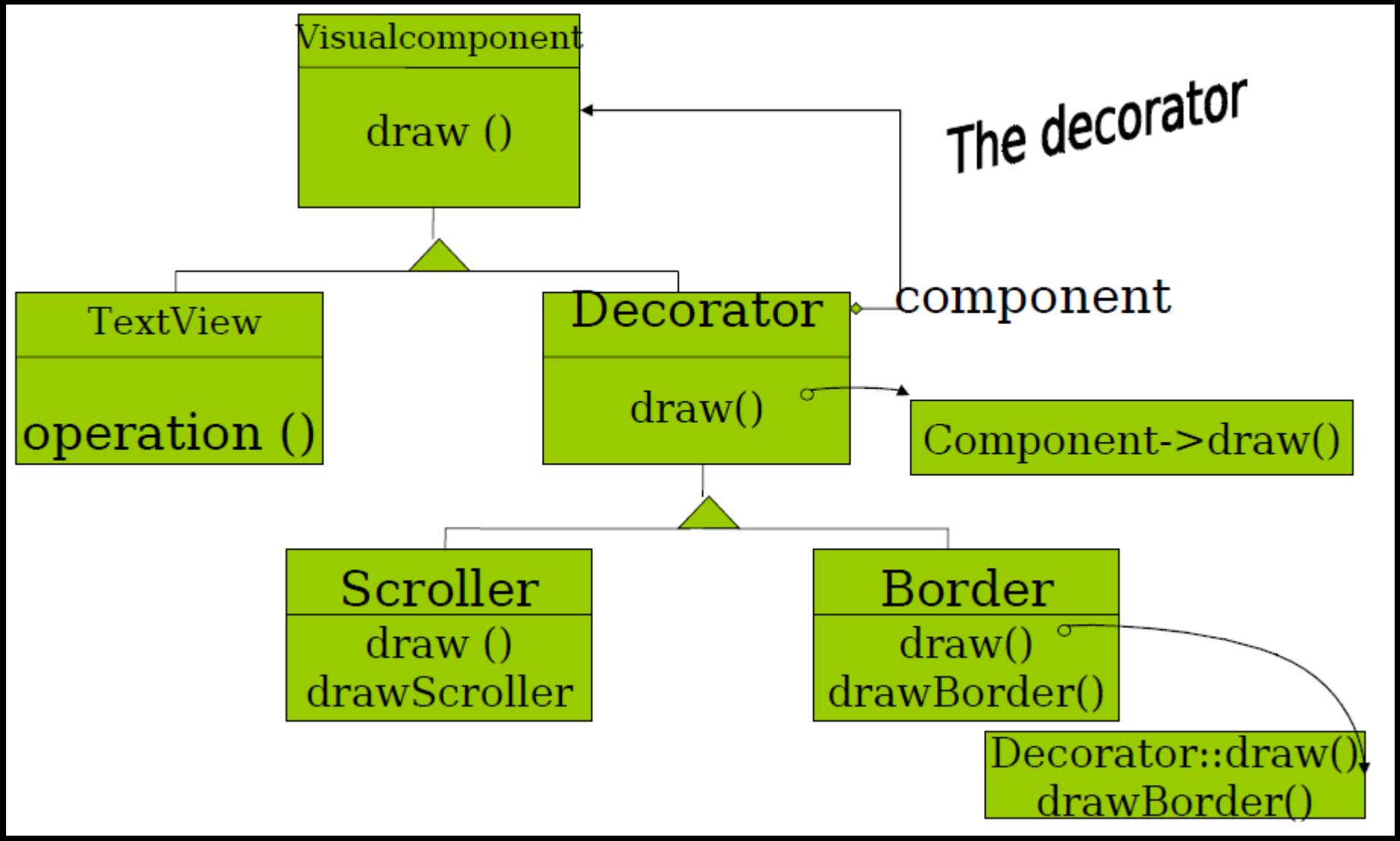
Classification of Patterns (Cont..)

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	**Factory Method**	**Adaptor Class**	**Interpreter** **Template Method**
	Object	Abstract Factory Builder Prototype Singleton	Adaptor (Object) Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

The Decorator: Object Diagram



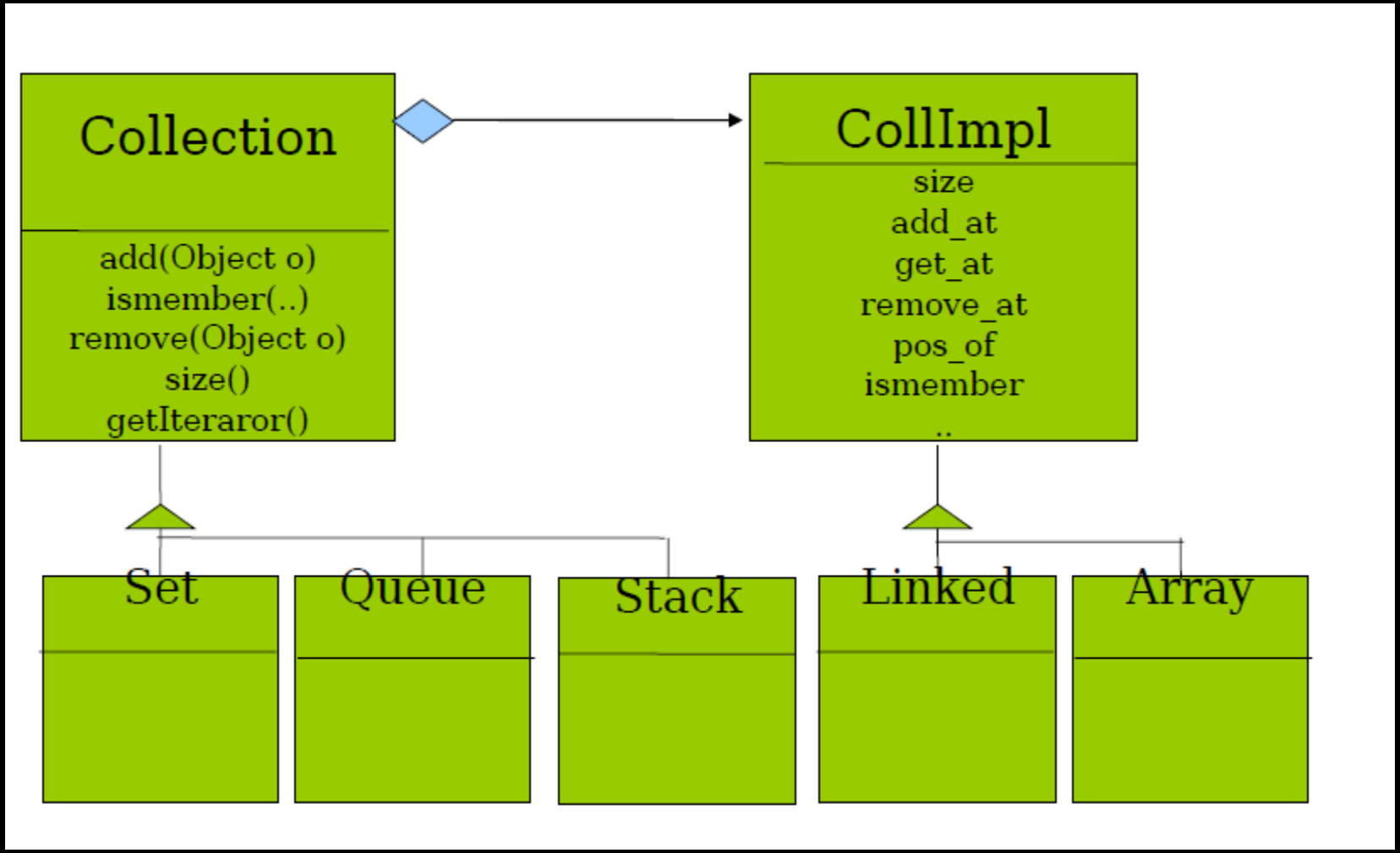
The Decorator



A problem: Varying Implementations and Abstractions

- In a collection hierarchy, on one hand, the abstractions vary
  - Collection – set, queue, stack
- On the other hand, their implementations may also vary
  - array based, linked list based

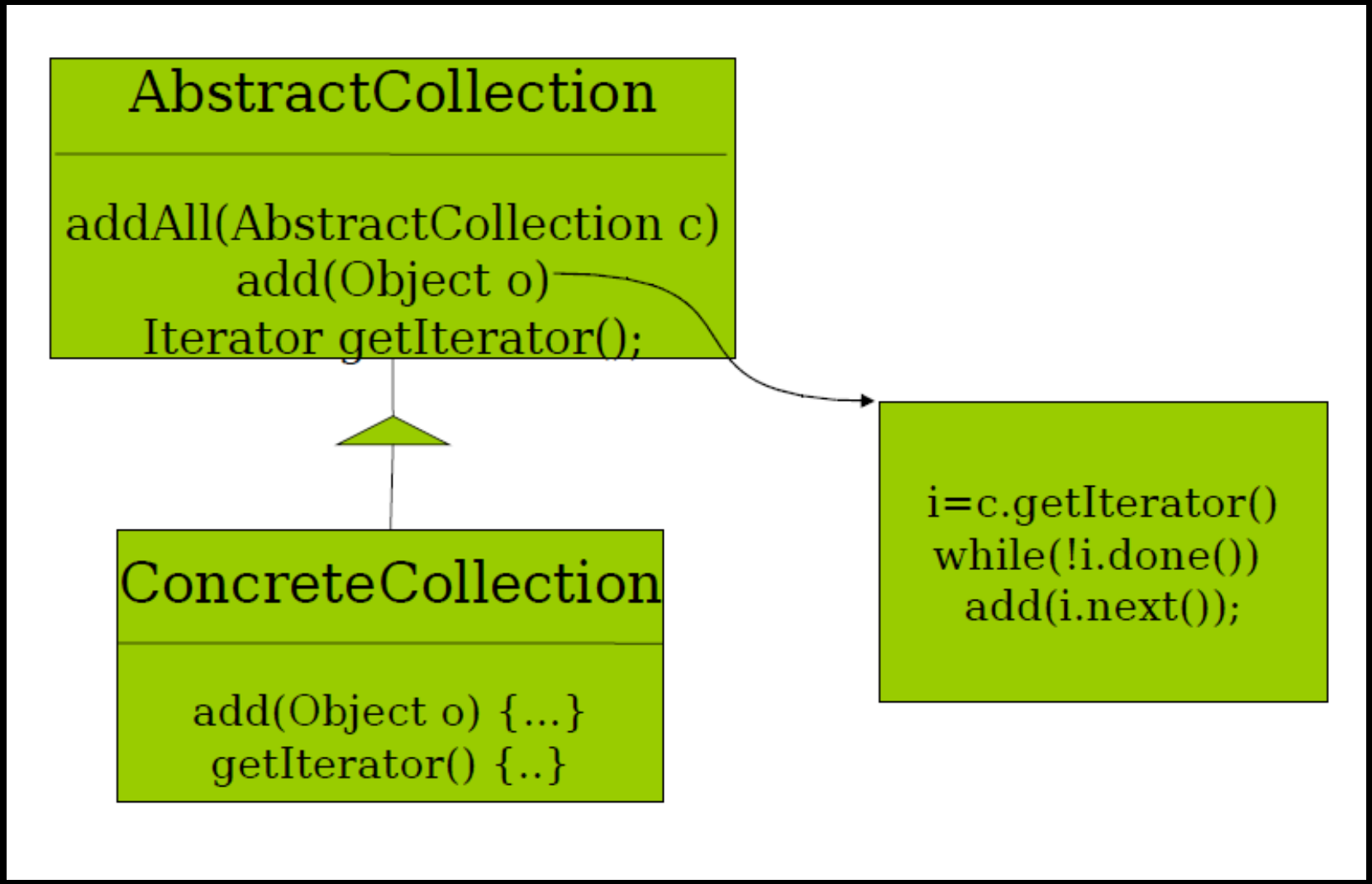
A structural pattern: Bridge



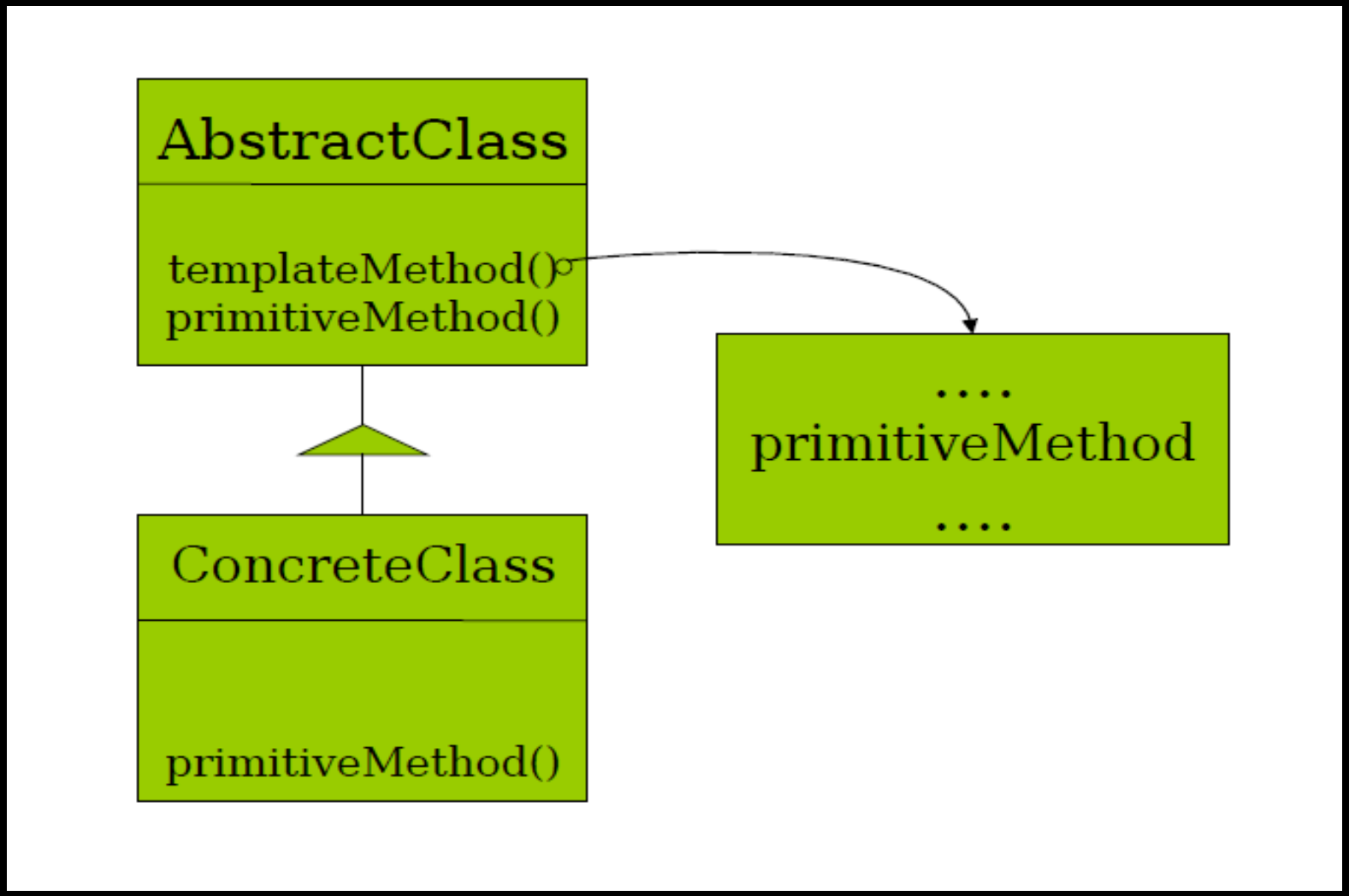
Next problem: Some steps in an implementation can vary

- In a hierarchy of classes, a behavior is quite common for all classes, but only that some steps are dependent on the nature of these classes
- How to avoid redundancy in defining such behavior
- Where should such a method be located?

In a Hierarchy: Template Method



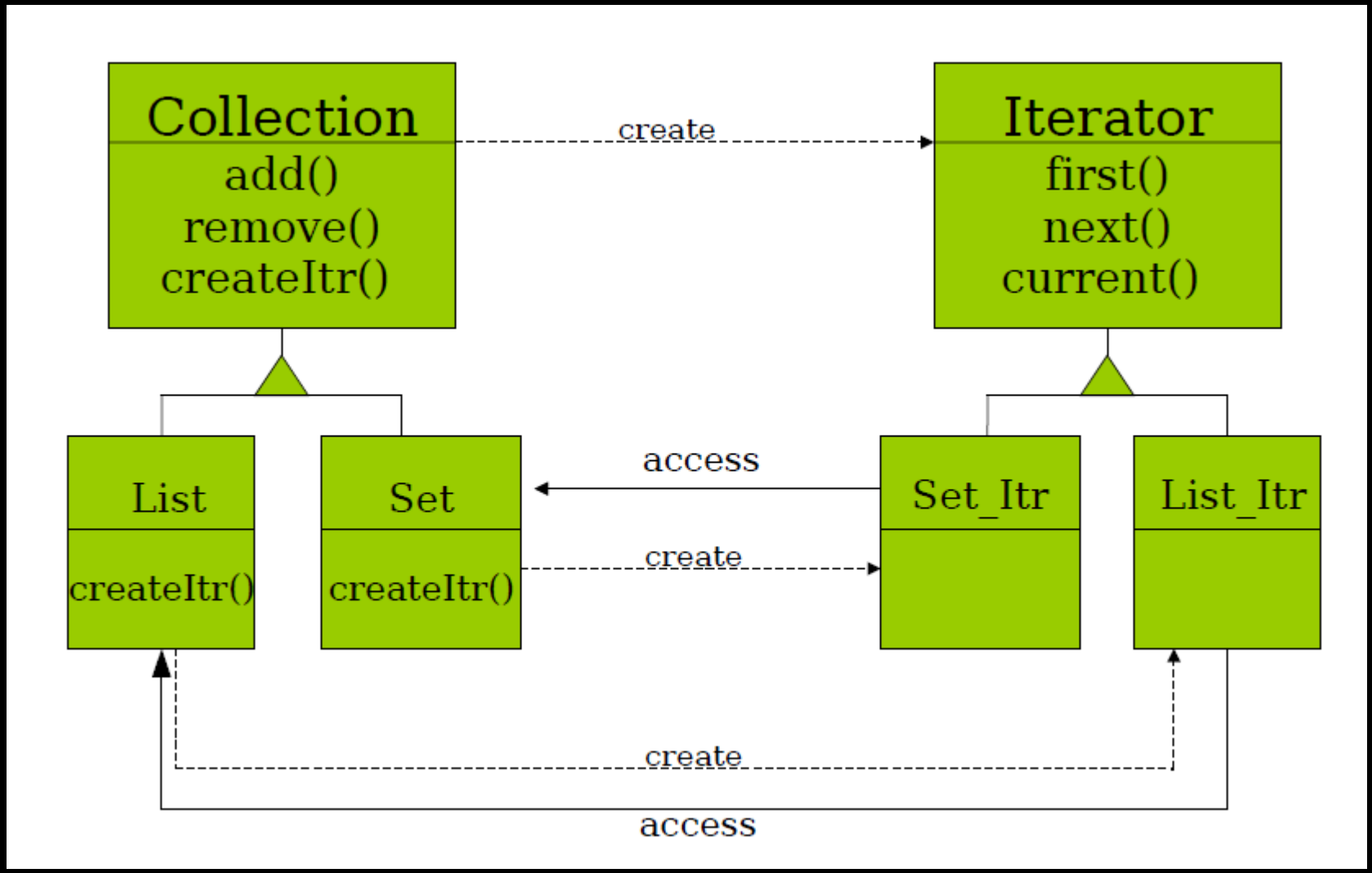
The Template Method Pattern



A Problem: Separate collection and iterations over the collection

- A collection supports members to add and remove elements, to check for membership
- Iterations over the collection can be specified separately such that the two abstractions are not intermingled
- The iterator hides the internal implementation of the collection
- Multiple ways of iteration can be supported
- Iterators can be used concurrently

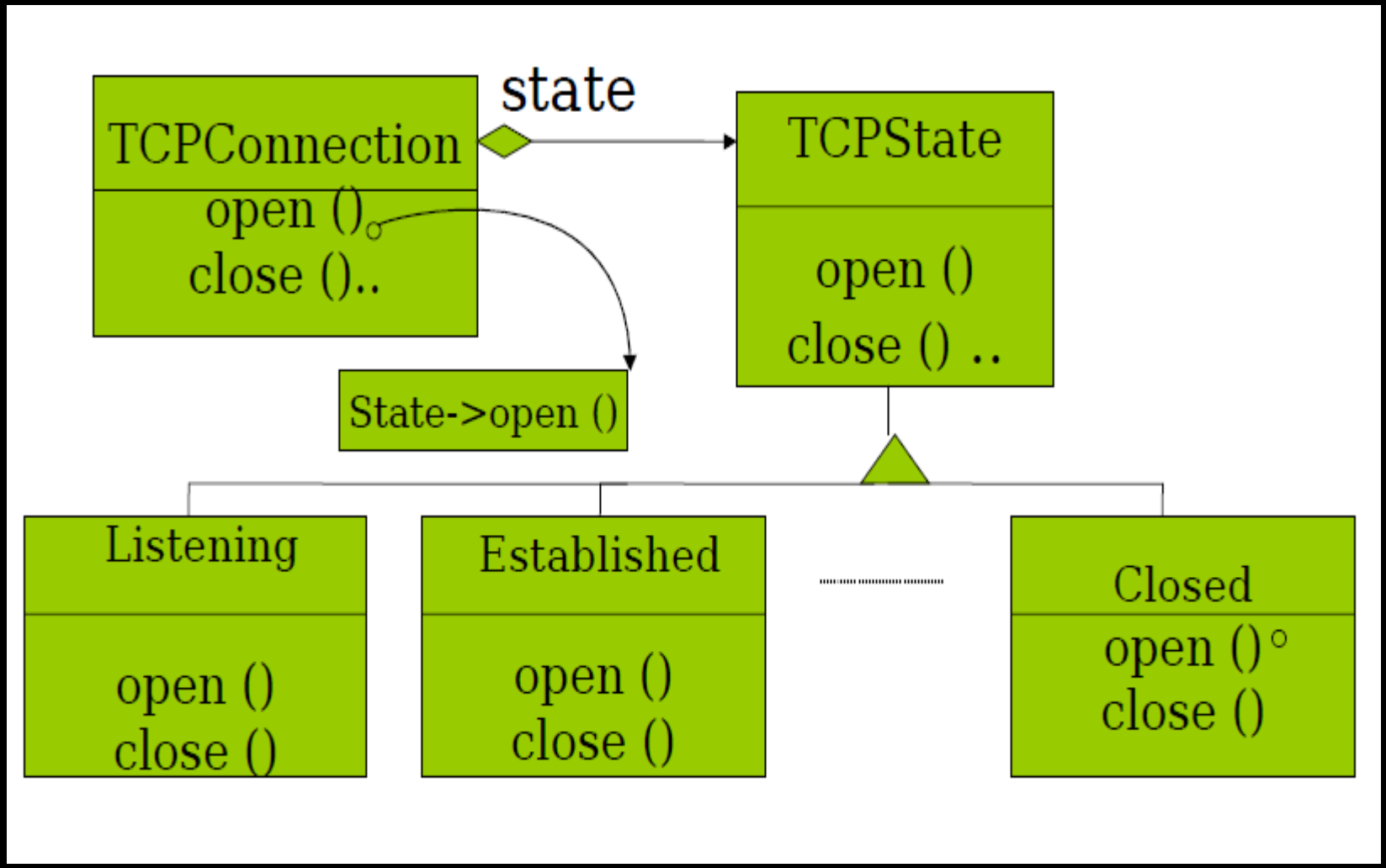
Iterator Pattern



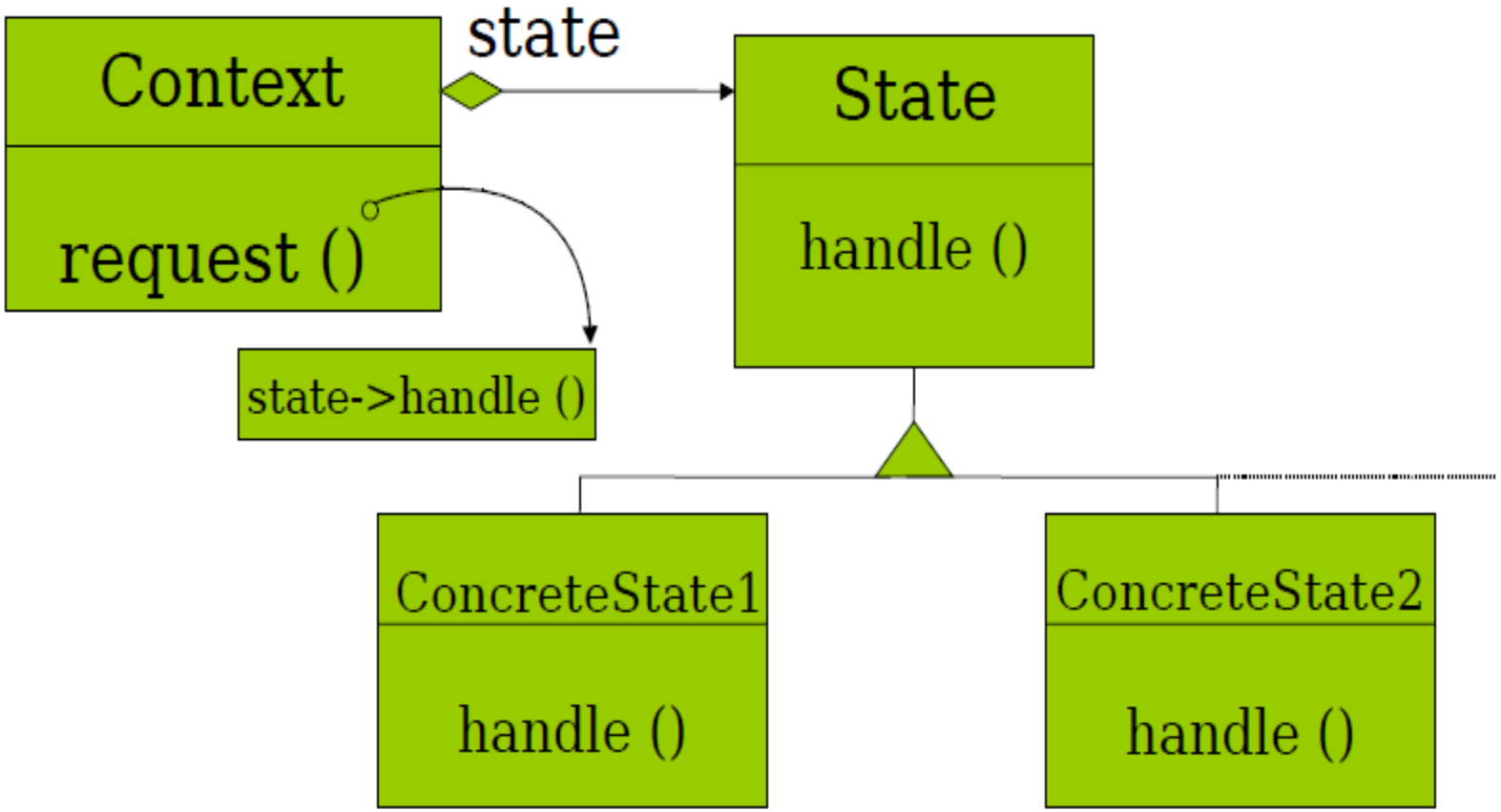
Next problem: An object alters its behavior as it changes its state

- Example:
  - A TCP connection object provides methods such as open(), close(), send()..
  - The connection object changes the behavior of these methods as it changes its state from disconnected to listening to established to closed

TCP States through the State Pattern



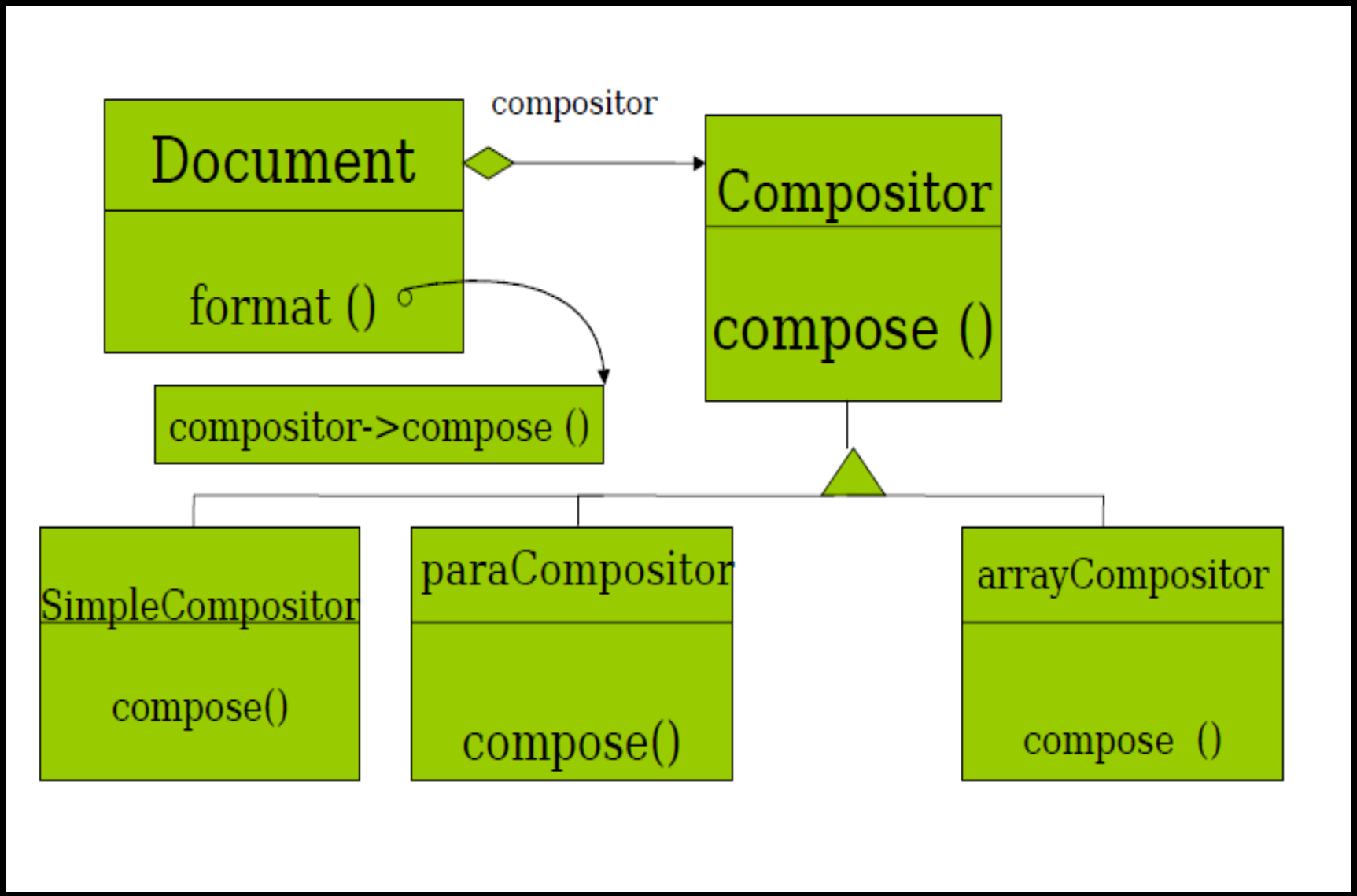
The State Pattern



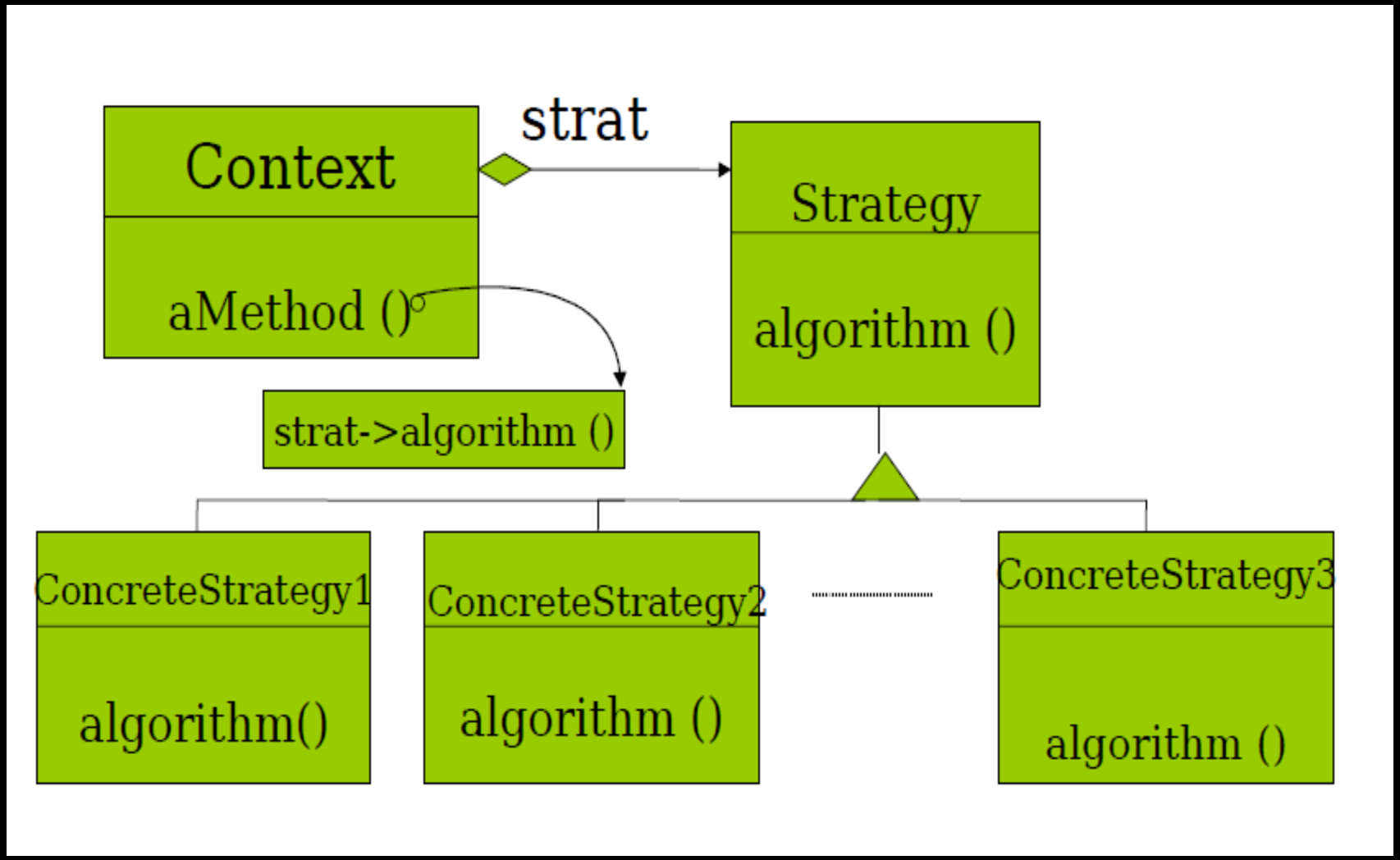
Next Problem: Use different algorithms for different situations in a given problem context: Strategy Pattern

- Example:
  - A document is composed of text. Various line breaking algorithms can be used in formatting the document before printing
- Consider the following strategies:
  - simple compose: determine line breaks, one line at a time
  - para compose: consider lines in an entire paragraph
  - array compose: each row has a fixed number of letters

The Solution



The Strategy Pattern



Thanks