* The State Pattern allows an object to have many different behaviors that are based on its internal state.
* Unlike a procedural state machine, the State Pattern represents state as a full-blown class.
* The context gets its behavior by delegating to the current state object it is composed with.
* By encapsulating each state into a class, we localize any changes that will need to be made.
* The State and Strategy Patterns have the same class diagram, but they different in intent.
* Strategy Pattern typically configures context classes with a behavior of algorithm.
* State Pattern allows a context to change its behavior as the state of the context changes.
* State transitions can be controlled by the state classes or by the context classes.
* Using the state pattern will typically result in a greater number of classes in your design.
* State classes may be shared among context classes.

ConcreteStateB

handle()

ConcreteStateA

handle()

state.handle()

State

handle()

Context

request()