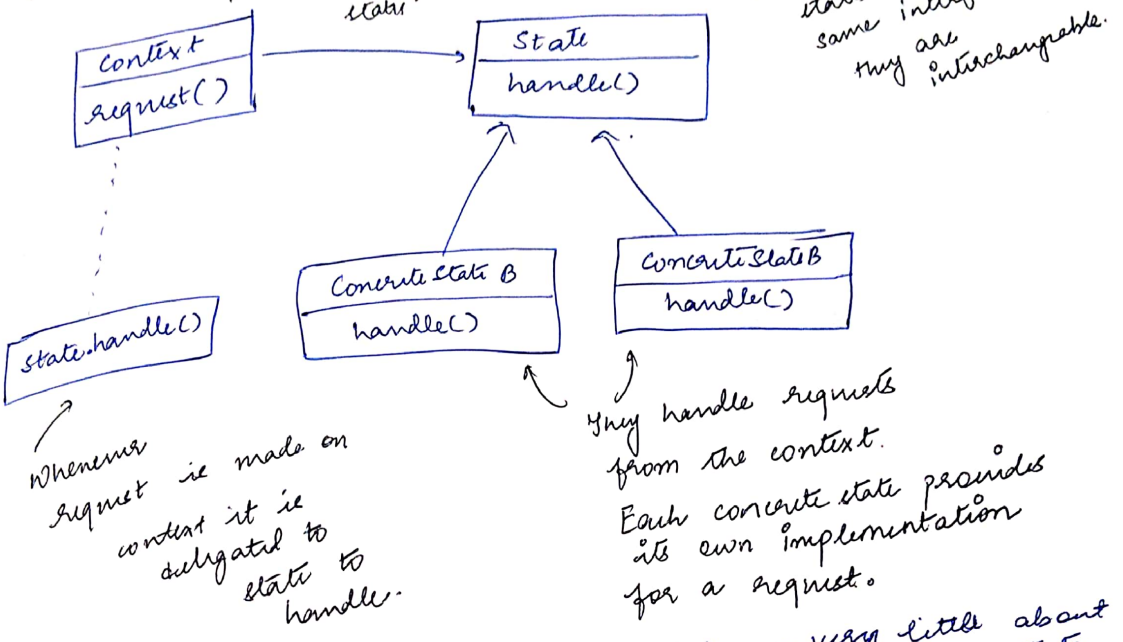
Chapter 10: State Pattern

* By structurally changing the implementation :
  - localized behaviour of each state into own class
  - removed 'if' statements
  - closed each state for modification

* The State pattern allows an object to alter its behaviour when its internal state changes. The object will appear to change its class.

* class diagram :



* Strategy vs the State pattern → client knows very little about state objects
  - client usually specifies the strategy object that the context is composed with.
  - alternative to putting lots of 'if'.

* Strategy
  - flexible alternative to subclassing
  - change behaviour by composing

* There are no dumb Qs.
  - Concrete States don't always decide what state to go next
  - Context is to decide on flow of state transitions.
  - client doesn't interact with states - Context.