Understanding State Design Pattern in complex, use below link <http://onjavahell.blogspot.in/2009/05/simple-example-of-state-design-pattern.html>

Both State and Strategy design patterns has similar structure, and both of them are based upon Open closed design principle, represents 'O' from [SOLID design principles](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html), they are totally different on there *intent*.  
State pattern helps object to manage state, while [Strategy pattern](http://java67.blogspot.sg/2012/09/top-10-java-design-pattern-interview-question-answer.html) allows client to choose different behaviour.  
Another difference, which is not easily visible is, who drives change in behaviour. In case of Strategy pattern, it's client, which provides different strategy to Context

On State pattern, state transition is managed by Context or State itself. Also, if you are managing state transition in State object itself, it must hold reference of Context e.g. Vending Machine, so that it can call setState() method to change current state of Context. On the other hand, Strategy object never held reference of Context, it's client which passes Strategy of there choice to Context.

Strategy design pattern in Java is used to encapsulate related set of algorithms to provide runtime flexibility to client. Client can choose any algorithm at runtime, without changing Context class, which uses Strategy object. Some of the popular example of Strategy pattern is writing code, which uses algorithms e.g. **encryption, compression or sorting algorithm**.  
  
State design pattern allows an object to behave differently at different state. e.g. a Vending Machine(), ATM Machine, DVD