1. **Agenda**:
   1. Let’s take a look at the steps that you need to follow when implementing State Design Pattern.
2. Text

   Description automatically generated **Steps**:
   1. Find distinct values of state in which our object can exist.  
      For example: If you have order object, it can be in state -> new state, paid state, in-transit state, delivered state.
   2. Each state value becomes a separate class.
   3. Each class will define implementation (behavior) which is specific to that value.  
      An order with state new has a different workflow when it is cancelled & different workflow when the order is paid state.
   4. Then in main/context class, we’re going to define methods being used by a client.  
      Object state is not exposed to the client. So client doesn’t know the current state of the object.  
      Our client is going to call methods from main/context class and those methods will delete the operation to whatever current state object they have with it.
   5. Then we have to decide how our state transition is going to happen.   
      In our order example, let’s say our order is in New State.   
      Then you have to decide how order moves from new state to paid state. Means context class will decide or the state itself decides.
3. The client is going to interact with main/context class and is unaware of existence of its state.  
   This gives this design pattern a power as we can **keep on adding new state thus a new behavior** and our client needs not to change because a new state is added.