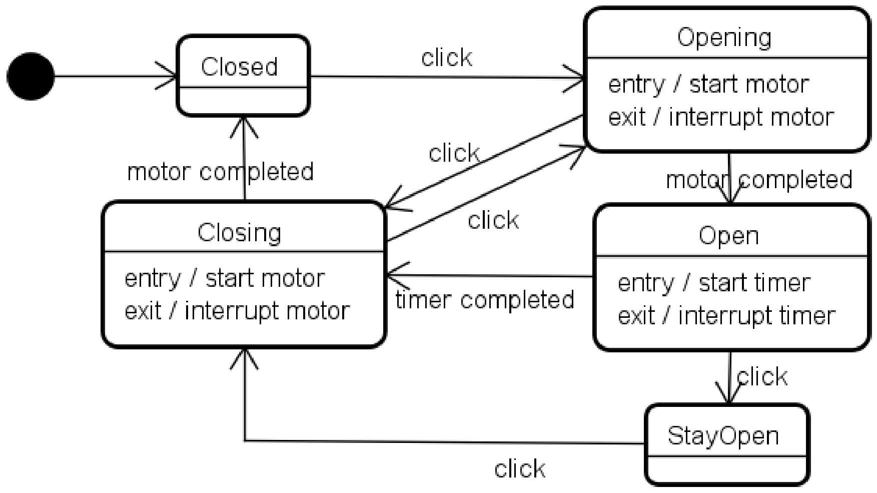
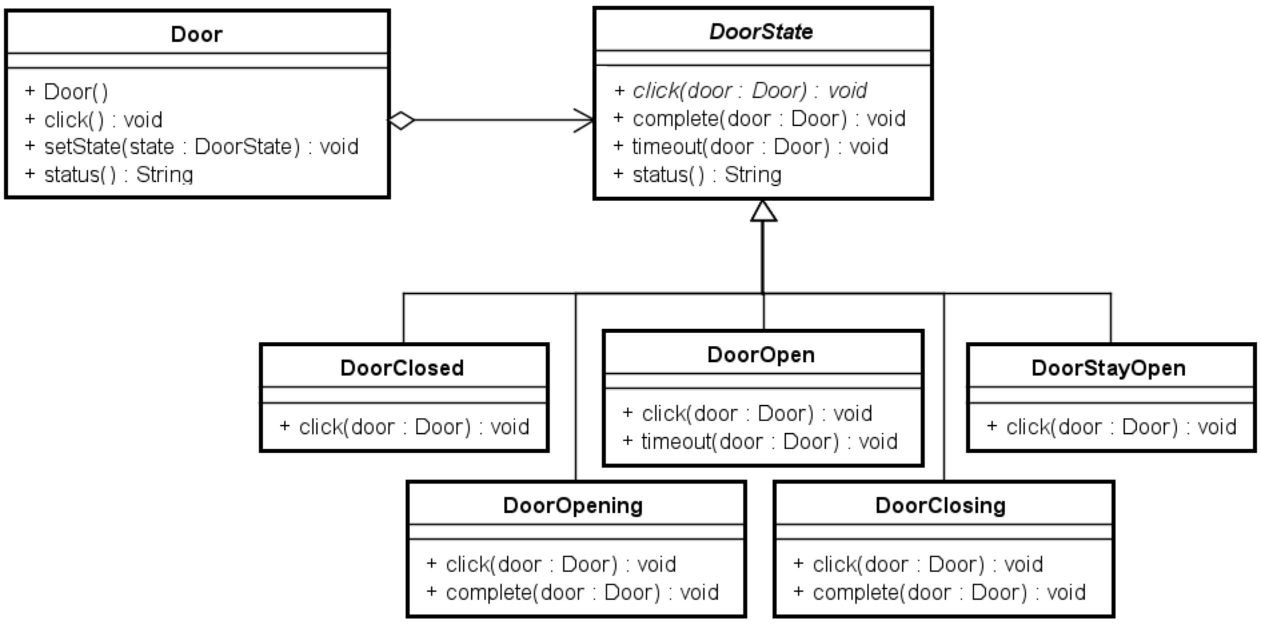
Exercise: State design pattern – Door (version 2)

Implement the state design pattern for the Door example given in the presentation (shown as version 2 in the presentation). This version creates a thread in states Opening and Closing to simulate time for the motor to complete the opening/closing (use e.g. 5 seconds as the sleep time) and a thread in the Open state to simulate a timeout (use e.g. 10 seconds). Class `Opening` is shown at the end of this exercise document, and can be used as a template for the other two classed.

You have to follow the UML state machine diagram and the UML class diagram shown below (*Note: Class Door do not have methods* `complete` *and* `timeout` *any longer)*:




Door
+ Door()
+ click() : void
+ setState(state : DoorState) : void
+ status() : String
DoorState
+ click(door : Door) : void
+ complete(door : Door) : void
+ timeout(door : Door) : void
+ status() : String
DoorClosed
+ click(door : Door) : void
DoorOpen
+ click(door : Door) : void
+ timeout(door : Door) : void
DoorStayOpen
+ click(door : Door) : void
DoorOpening
+ click(door : Door) : void
+ complete(door : Door) : void
DoorClosing
+ click(door : Door) : void
+ complete(door : Door) : void

Insert print statements in methods `click` and `setState` in class `Door` to see the current state (call status for the state)

Test it in a main method in which you create a Door object and call click in different stated and also test that you get a timeout or completed motor opening or closing the door (using sleep in the main method)

Class DoorOpening

```
public class DoorOpening extends DoorState
{
  private Thread motor;
  private boolean completed;

  public DoorOpening(Door door)
  {
    completed = false;
    motor = new Thread(() -> {
      try
      {
        Thread.sleep(5000);
        complete(door);
      }
      catch (InterruptedException e)
      {
        System.out.println("Motor interrupted (opening)");
      }
    });
    motor.start();
  }

  private synchronized void complete(Door door)
  {
    if (!completed)
    {
      System.out.println("Motor ended (opening)");
      door.setState(new DoorOpen(door));
      completed = true;
    }
  }

  @Override public synchronized void click(Door door)
  {
    if (!completed)
    {
      motor.interrupt();
      door.setState(new DoorClosing(door));
      completed = true;
    }
  }
}
```