Homework12

This Java program uses the State Pattern to simulate the air ticket system. The AirTicket class is the context that holds the current state, and the states (Available, Locked, and Sold) handle the actions and state transitions according to the state diagram.

```
//State Interface:

interface AirTicketState {
    void lock();
    void unlock();
    void buy();
    void assignTo();
    void timeOut();
    void exchange();
}

// Concrete States:
// Available State:
class Available implements AirTicketState {
    private AirTicket airTicket;

    public Available(AirTicket airTicket) {
        this.airTicket = airTicket;
    }

    @Override
    public void lock() {
        System.out.println("Ticket locked.");
        airTicket.setState(airTicket.getLockedState());
    }

    @Override
    public void unlock() {
        // Not applicable in the Available state
        System.out.println("Ticket is already available.");
    }

    @Override
    public void buy() {
        // Not applicable in the Available state
        System.out.println("Ticket is not locked. Please lock it first.");
    }

    @Override
    public void assignTo() {
```

```java
        // Not applicable in the Available state
        System.out.println("Ticket is not locked. Please lock it first.");
    }

    @Override
    public void timeOut() {
        // Not applicable in the Available state
        System.out.println("Ticket is not locked. Please lock it first.");
    }

    @Override
    public void exchange() {
        // Not applicable in the Available state
        System.out.println("Ticket is not locked. Please lock it first.");
    }
}

//Locked State:
class Locked implements AirTicketState {
    private AirTicket airTicket;

    public Locked(AirTicket airTicket) {
        this.airTicket = airTicket;
    }

    @Override
    public void lock() {
        // Not applicable in the Locked state
        System.out.println("Ticket is already locked.");
    }

    @Override
    public void unlock() {
        System.out.println("Ticket unlocked.");
        airTicket.setState(airTicket.getAvailableState());
    }

    @Override
    public void buy() {
        System.out.println("Ticket bought.");
        airTicket.setState(airTicket.getSoldState());
    }

    @Override
    public void assignTo() {
        // Not applicable in the Locked state
        System.out.println("Ticket is locked. Please buy it first.");
    }
    @Override
    public void timeOut() {
        System.out.println("Ticket unlocked due to timeout.");
        airTicket.setState(airTicket.getAvailableState());
    }

    @Override
    public void exchange() {
        // Not applicable in the Locked state
        System.out.println("Ticket is locked. Please buy it first.");
    }
}


//Sold State:
class Sold implements AirTicketState {
    private AirTicket airTicket;

    public Sold(AirTicket airTicket) {
        this.airTicket = airTicket;
    }

    @Override
    public void lock() {
        // Not applicable in the Sold state
        System.out.println("Ticket is sold. Cannot lock.");
    }

    @Override
    public void unlock() {
        // Not applicable in the Sold state
        System.out.println("Ticket is sold. Cannot unlock.");
    }

    @Override
    public void buy() {
        // Not applicable in the Sold state
        System.out.println("Ticket is already sold.");
    }

    @Override
    public void assignTo() {
        // Not applicable in the Sold state
        System.out.println("Ticket is sold. Cannot assign.");
    }

    @Override
    public void timeOut() {
        // Not applicable in the Sold state
        System.out.println("Ticket is sold. Cannot timeout.");
    }

    @Override
    public void exchange() {
        System.out.println("Ticket exchanged.");
        airTicket.setState(airTicket.getLockedState());
    }
}


// Context Class:

class AirTicket {
    private AirTicketState availableState;
    private AirTicketState lockedState;
    private AirTicketState soldState;
    private AirTicketState currentState;

    public AirTicket() {
        availableState = new Available(this);
        lockedState = new Locked(this);
        soldState = new Sold(this);
        currentState = availableState;
    }

    public void lock() {
        currentState.lock();
    }

    public void unlock() {
        currentState.unlock();
    }

    public void buy() {
        currentState.buy();
    }

    public void assignTo() {
        currentState.assignTo();
    }

    public void timeOut() {
        currentState.timeOut();
    }

    public void exchange() {
        currentState.exchange();
    }
```

```java
    public AirTicketState getAvailableState() {
        return availableState;
    }

    public AirTicketState getLockedState() {
        return lockedState;
    }

    public AirTicketState getSoldState() {
        return soldState;
    }

    public void setState(AirTicketState state) {
        this.currentState = state;
    }
}


//Test the Air Ticket System:

public class Main {
    public static void main(String[] args) {
        AirTicket ticket = new AirTicket();
        ticket.lock();
        ticket.buy();
        ticket.exchange();
        ticket.unlock();
    }
}
```