Practical on the State Pattern

In this practical you are going to take the model answer to the MVC practical (ie Traffic Lights using the Observer Pattern) and adapt it to use the State Pattern as well. This will eliminate the messy assignment statements and complex case analysis that currently would make it more difficult to make changes such as, for example, changing the order of traffic lights. The class diagram in the lecture notes may make it easier for you to understand what you are doing.

1. Create a new abstract class called `State` with code as shown at the bottom.
2. Create subclasses `RedState`, `GreenState`, `AmberState` and `RedAmberState`. For each class, override the constructor to call the superclass constructor and implement the four abstract methods. Consult the notes to find out how to change the state using the getters and setters of `TrafficLightModel`.
3. Rewrite the `TrafficLightModel` methods `change`, `getRed`, `getAmber`, and `getGreen` so that they delegate to a state object, which you should declare as an instance variable.
4. Add instance variables for the four states and implement their getters, so that `State` can call them.
5. Implement the setter for the state instance variable.
6. Rewrite initialise to create the four states and set state to the instance of `RedState`.
7. Review the steps above to check your understanding of the State Pattern. Then compile and run your code.

Initial Code for the State Pattern

```
public abstract class State {
     public TrafficLightModel model;
     public State(TrafficLightModel model){
       this.model = model;
     }
     public abstract boolean getAmber();
     public abstract boolean getGreen();
     public abstract boolean getRed();
     public abstract void change();
   }
```