State Design Pattern

by Raul TRONCIU

What is State DP

In State pattern a class behaviour changes based on its state.

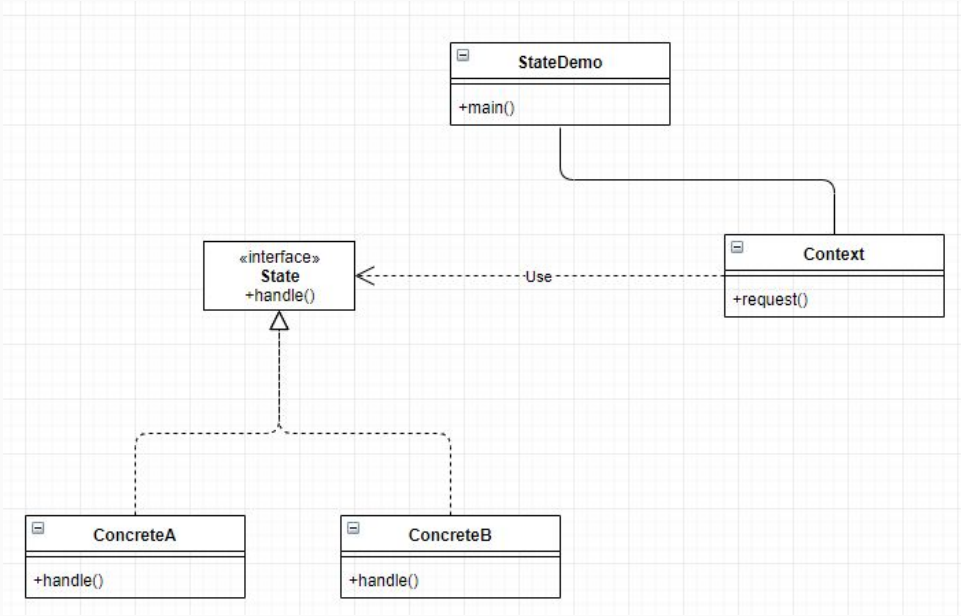
In State pattern we create objects which represent various states and a context object whose behaviour varies as its state object changes.

How to implement

We start by creating a *State* interface defining an action and concrete state classes implementing the *State* interface.

Context is a class which carries a state and changes its behaviour depending on that state.

State pattern Diagram



Example

```
interface MobileAlertState { public void alert(AlertStateContext ctx); }

class AlertStateContext  {

    private MobileAlertState currentState;

    public AlertStateContext() { currentState = new Vibration(); }

     public void setState(MobileAlertState state) { currentState = state; }

    public void alert() { currentState.alert(this); }

}
```

```
class Vibration implements MobileAlertState {
    @Override
    public void alert(AlertStateContext ctx) {
        System.out.println("vibration...");
    }

}

class Silent implements MobileAlertState {
    @Override
    public void alert(AlertStateContext ctx) {
        System.out.println("silent...");
    }
}
```

Example (cnt.)

```
class StatePattern  {

   public static void main(String[] args)  {

      AlertStateContext stateContext = new AlertStateContext();

      stateContext.alert();

      stateContext.alert();

      stateContext.setState(new Silent());

      stateContext.alert();

      stateContext.alert();

      stateContext.alert();

   }

}
```

Example (cnt.)

Output:

```
vibration...

vibration...

silent...

silent...

silent...
```

Advantages

- Benefits of implementing a polymorphic behaviour
- Easy to add more states to support more behaviours
- Improves Cohesion since state-specific behaviors are aggregated into the ConcreteState classes

Disadvantage

- The number of classes grows fast