

The State Pattern

Human Computer Interaction Research
University of Nevada, Reno



Iterator Pattern

★ Structural Patterns

- » adapter
- » façade
- » composite

★ Creational Patterns

- » factory method
- » abstract factory
- » singleton

★ Behavioral Patterns

- » strategy
- » observer
- » decorator
- » command
- » template method
- » iterator
- » **state**

Problem

Methods that have large, multipart conditional statements that depend on the object's state are difficult to maintain and extend

example

```
static int A=1;
static int B=2;
static int C=3;

private int state;

void DoSomething() {
    if (state==A) {
        // do this
    }
    else if (state==B) {
        state=A;
    }
    else if (state==C) {
        state=A;
    }
}
```

**"Classes should be open for extension,
but closed for modification"**

```
void DoThis() {
    if (state==A) {
        // do this
    }
    else if (state==B) {
        state=A;
    }
    else if (state==C) {
        state=A;
    }
}
```

works but difficult to extend

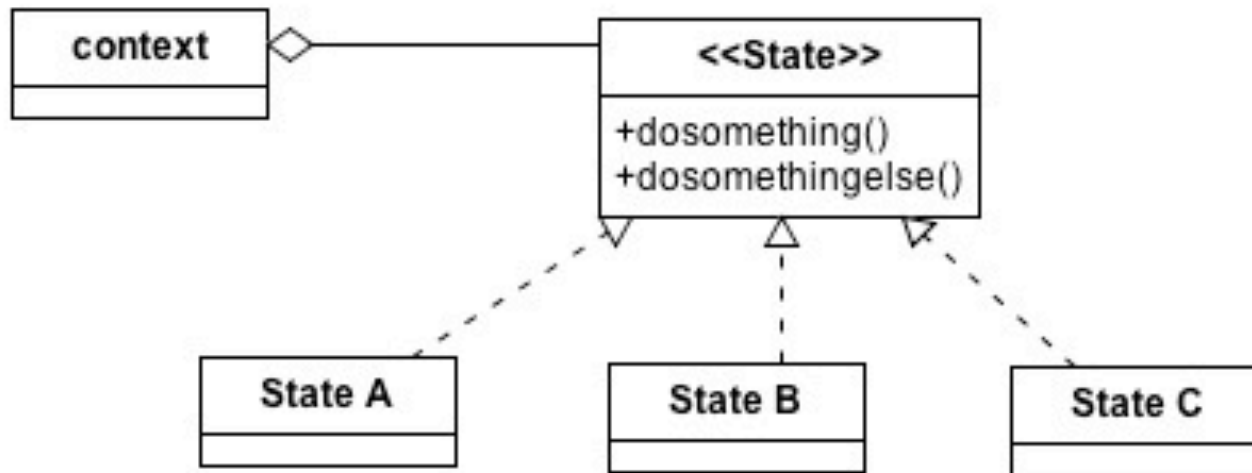
```
static int D=4;
```

Human Computer Interaction Research
University of Nevada, Reno

State Pattern

The State Pattern allows an object to alter its behavior when its internal state changes. The object will appear to change its class.


Class diagram



**Looks like
Strategy but
Intent is different**

State Interface

```
public interface State {  
    public void doThis();  
    public void doThat();  
}
```



```
public class State_A implements State {  
    StateContext statecontext;  
  
    public State_A(StateContext statecontext) {  
        this.statecontext=statecontext;  
    }  
  
    public void doThat() {  
        statecontext.setState(statecontext.getState_B());  
    }  
  
    public void doThis() {  
    }  
  
}
```

```
public class StateContext {
    State State_A;
    State State_B;
    State myState;

    public StateContext() {
        State_A = new State_A(this);
        State_B = new State_B(this);
        myState = State_A;
    }
    public void setState(State stateName) {
        this.myState = stateName;
    }

    public State getState_A() {
        return State_A;
    }

    public State getState_B() {
        return State_B;
    }
}
```

exercise

