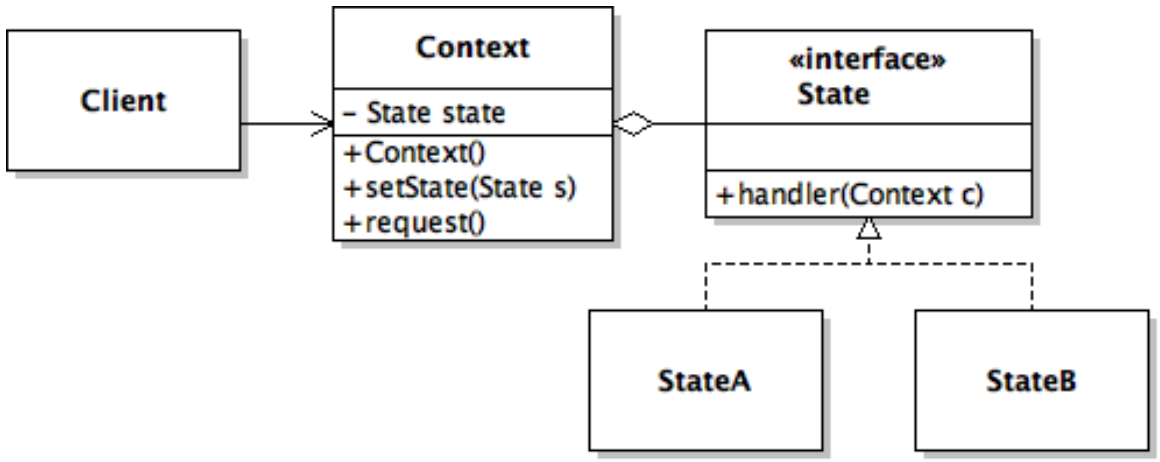
# CECS 277 – Lecture 24 – Design Patterns

**State**

The State pattern is a Behavioral design pattern that allows an object to alter its behavior when its state changes. It uses inheritance to allow subclasses to represent different states and functionality that can change during runtime. This is an easy way for an object to change its class type while the program is running. The State pattern is commonly used to represent the different states of a state machine.

**State UML –** The State pattern has an interface with methods that each implementing State class will need to override. The Context class has a State object that is initially set in its constructor. The request method is used to call the handler methods from whichever state it is currently in. The State handler method is the code that needs to happen when the object is in that state, after which it calls setState if the state needs to be changed.




**State Template Code:**

```
public interface State {
   public void handler( Context c );
}
public class StateA implements State {
   @Override public void handler( Context c ) {
      System.out.println( "State = A" );
      c.setState( new StateB( ) );
   }
}
public class StateB implements State {
   @Override public void handler( Context c ) {
      System.out.println( "State = B" );
      c.setState( new StateA( ) );
   }
}
public class Context {
   private State state;
   public Context( ) {
```

```
      state = new StateA( );
   }
   public void setState( State s ) {
      state = s;
   }
   public void request( ) {
      state.handler( this );
   }
}
public class TestState {
   public static void main( String [] args ) {
      Context c = new Context( );
      c.request( );  //State = A
      c.request( );  //State = B
      c.request( );  //State = A
   }
}
```

**State Example Code:**

```
public interface RadioState {
   public void turnOn( Radio r );
   public void turnOff( Radio r );
   public void changeStation( Radio r );
}
public class OffState implements RadioState {
   @Override public void turnOn( Radio r ) {
      System.out.println( "Turning on radio..." );
      System.out.println( "Station -> Rock );
      r.setState( new RockState( ) );
   }
   @Override public void turnOff( Radio r ) {
      System.out.println( "Radio is already off..." );
   }
   @Override public void changeStation( Radio r ) {
      System.out.println( "Turn the radio on first..." );
   }
}
public class RockState implements RadioState {
   @Override public void turnOn( Radio r ) {
      System.out.println( "Radio is already on..." );
   }
   @Override public void turnOff( Radio r ) {
      System.out.println( "Turning off radio..." );
      r.setState( new OffState( ) );
   }
   @Override public void changeStation( Radio r ) {
      System.out.println( "Station -> Oldies" );
```

```
      r.setState( new OldiesState( ) );
   }
}
public class OldiesState implements RadioState {
   @Override public void turnOn( Radio r ) {
      System.out.println( "Radio is already on..." );
   }
   @Override public void turnOff( Radio r ) {
      System.out.println( "Turning off radio..." );
      r.setState( new OffState( ) );
   }
   @Override public void changeStation( Radio r ) {
      System.out.println( "Station -> Pop" );
      r.setState( new PopState( ) );
   }
}
public class PopState implements RadioState {
   @Override public void turnOn( Radio r ) {
      System.out.println( "Radio is already on..." );
   }
   @Override public void turnOff( Radio r ) {
      System.out.println( "Turning off radio..." );
      r.setState( new OffState( ) );
   }
   @Override public void changeStation( Radio r ) {
      System.out.println( "Station -> Rock" );
      r.setState( new RockState( ) );
   }
}
public class Radio {
   private RadioState state;
   public Radio( ) {
      state = new OffState( );
   }
   public void setState( RadioState s ) {
      state = s;
   }
   public void offSwitch( ) {
      state.turnOff( this );
   }
   public void onSwitch( ) {
      state.turnOn( this );
   }
   public void channelSwitch( ) {
      state.changeStation( this );
   }
}
```

```
public class Client {
   public static void main( String [] args ) {
      Radio r = new Radio( );
      r.channelSwitch( );
      r.onSwitch( );
      r.channelSwitch( );
      r.channelSwitch( );
      r.channelSwitch( );
      r.offSwitch( );
   }
}
/*Output:
   Turn the radio on first...
   Turning on radio...
   Station -> Rock
   Station -> Oldies
   Station -> Pop
   Station -> Rock
   Turning off radio...
*/
```