State Design Pattern

Discussion

In State pattern a class behavior changes based on its state. This type of design pattern comes under behavior pattern.

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

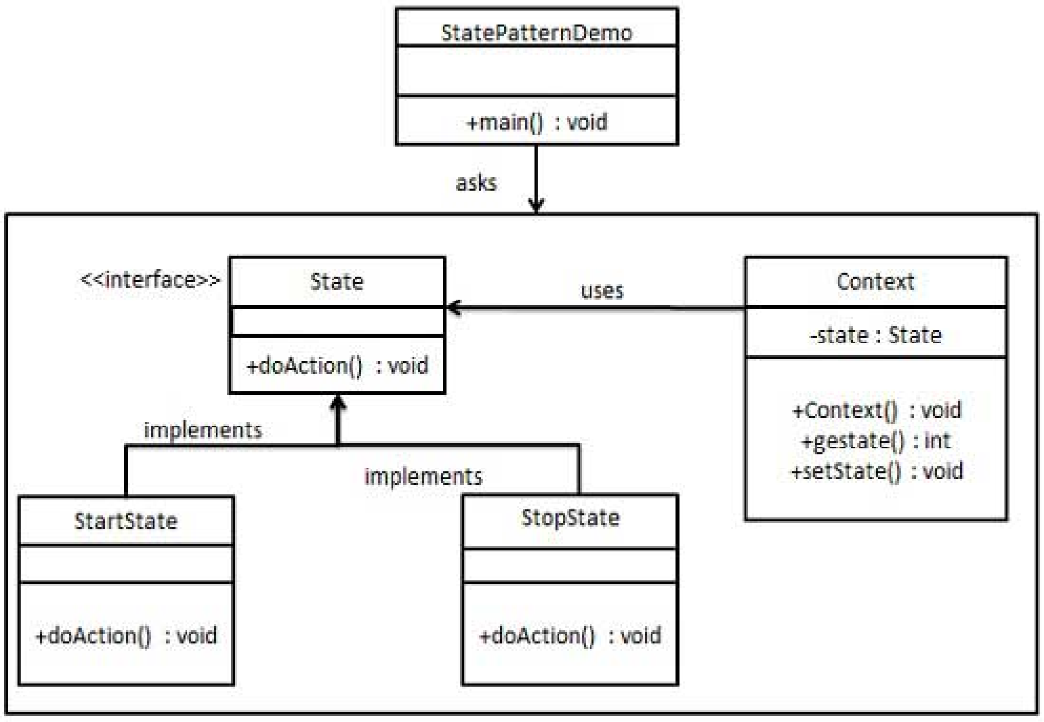
We are going to create a State interface defining an action and concrete state classes implementing the State interface. Context is a class which carries a State.

StatePatternDemo, our demo class, will use Context and state objects to demonstrate change in Context behavior based on type of state it is in.

Example

Consider ATM Machine. For simplicity two states of ATM are considered Initializing, and Running.

Diagram:



Implementation:

~~Class~~/ Interface: State

// Create the Interace which has doAction() functionality

```
package StatePat;

public interface State {
       public void doAction(Context context);
}
```

Class: Context

//Create the Context which will take care of execution of State functionality by setting the state

```
package StatePat;

public class Context {
         private State state;

         public Context(){
              state=null;

         }

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

         public State getState(){
            return state;
         }
      }
```

Class: Init // initializing state

//Create Concrete Class Init implementing interface
//here the concrete class is any state of system which you have considered for this implementation

```
package StatePat;

public class Init implements State {

         public void doAction(Context context) {
            System.out.println("ATM is in Initializing State");
            context.setState(this);
         }
```

```
          public String toString(){
             return "Init State";
          }
       }
```

Class: Runing // Running State

```
//Create Concrete Class Running implementing interface
//here the concrete class is any state of system which you have considered for this
implementation

package StatePat;

public class Runing implements State {

          public void doAction(Context context) {
             System.out.println("ATM is in Running state");
             context.setState(this);
          }

          public String toString(){
             return "Running State";
          }
       }
```

Class: StateDemo

```
//Use the Context to see change in behaviour when State changes
// Here you have to first initialize context.

package StatePat;

public class StateDemo {

   public static void main(String[] args) {

// Here you have to first initialize context (Create context object).
            Context context = new Context();
// here you can use events to go in particular system state by using
//if(condition/event)

             Init InitState = new Init();
             InitState.doAction(context); // the action will be performed on the
basis of current state

             System.out.println(context.getState().toString());

             Runing RunState = new Runing();
             RunState.doAction(context);

             System.out.println(context.getState().toString());
          }
       }
```

Output:

```
ATM is in Initializing State
Init State
ATM is in Running state
Running State
```