Centro Universitário UniBH
Ciência da Computação
Práticas de Programação
Professor: Lucas Schmidt

Aula Prática - 20 Padrões de Comportamento - State

Intenção

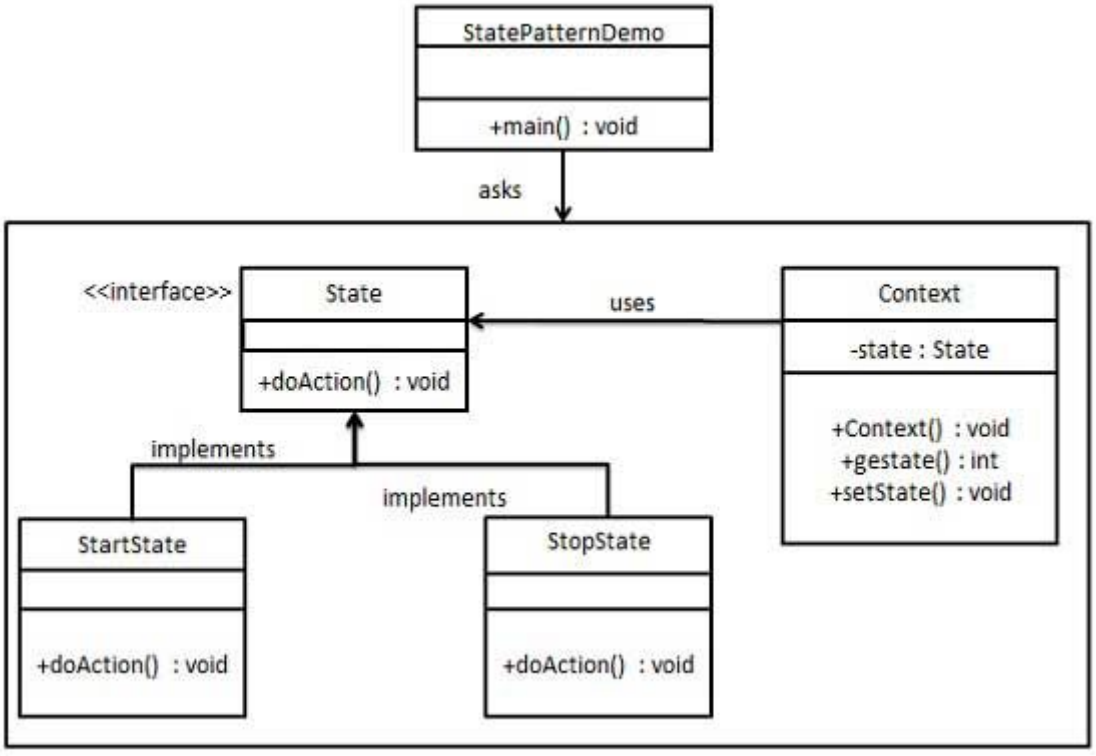
Permitir que um objeto altere seu comportamento quando muda de estado interno. O objeto aparenta mudar de classe. Também conhecido como: Objects for States

Usar este padrão quando…

- O comportamento de um objeto depende do seu estado, que é alterado em tempo de execução;
- Operações de um objeto possuem condicionais grandes e com muitas partes (sintoma do caso anterior).

Vantagens e desvantagens

- Separa comportamento dependente de estado:
  - Novos estados/comportamentos podem ser facilmente adicionados.
- Transição de estados é explícita:
  - Fica claro no diagrama de classes os estados possíveis de um objeto.
- States podem ser compartilhados:
  - Somente se eles não armazenarem estado em atributos.



Passo 1
Crie uma interface.

State.java

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

Passo 2
Crie classes concretas implementando a mesma interface.

StartState.java

```
public class StartState implements State {

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

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

StopState.java

```
public class StopState implements State {

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

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

Passo 3
Criar classe de contexto.

Context.java

```
public class Context {
   private State state;

   public Context(){
      state = null;
   }

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

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

Passo 4
Use o contexto para ver a mudança no comportamento quando o estado muda.

StatePatternDemo.java

```
public class StatePatternDemo {
   public static void main(String[] args) {
      Context context = new Context();

      StartState startState = new StartState();
      startState.doAction(context);

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

      StopState stopState = new StopState();
      stopState.doAction(context);

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

Passo 5

Teste sua implementação!