State Pattern

Design Patterns

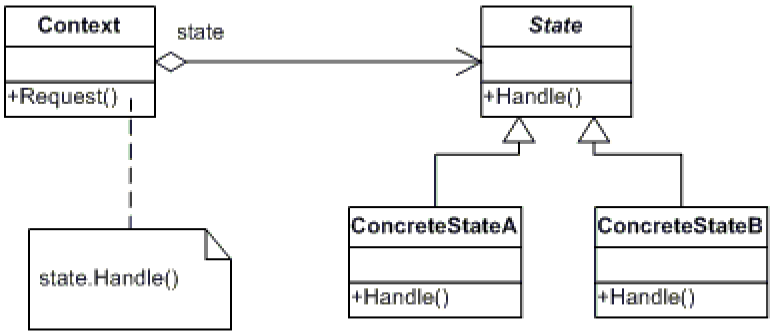
Christopher Doseck

11/30/2016

## Introduction

This assignment is an application that I created to show how the state pattern works. In this application I am using hot chocolate as a demonstration. Each of the temperatures of the hot chocolate is a different state.

## The UML Diagram for State



The UML Diagram for the state pattern, shown on the right, shows the classes that are needed to have the requirements. The Context class is the context class from the diagram. The State interface is the interface, and the temperatures are the ConcreteStates. The table below shows how all of the classes were used.

| | |
|---|---|
| Context | This is the Context class shown in the UML Diagram. |
| State | This is the State interface shown in the UML Diagram. All of the concrete states derive from it. |
| Burning, Hot, Warm, Cold, and Empty | These are all individual ConcreteState classes. They derive from the State interface. |
| Form1 | This is the client that shows the state pattern in action. |

## Narrative

```
public interface State
{
    string getTemp();
    string getDrinkMessage();
    State getNextState();
}
```

This is the State interface. It shows the required methods for each of its children.

```
internal class Warm : State
{
    public string getDrinkMessage()
    {
        return "This drink is nice and warm, it is the perfect temperature to enjoy.";
    }

    public State getNextState()
    {
        return new Cold();
    }

    public string getTemp()
    {
        return "Warm";
    }
}
```

This is the Warm class. It inherits from the State interface.

The getDrinkMessage() method and the getTemp() method return strings for the message to be displayed when drank and to display the temperature.

The getNextState() method returns the state after the current state. These go from Burning to Hot to Warm to Cold. Cold returns Cold and Empty returns Empty because neither of these things change by waiting.

The rest of the ConcreteStates are not included because they are not necessary.

```
public class Context
{
    State state;
    public Context()
    {
        state = new Burning();
    }

    public string getTemp()
    {
        return state.getTemp();
    }

    public string getMessage()
    {
        return state.getDrinkMessage();
    }

    public void nextState()
    {
        state = state.getNextState();
    }

    public void drankCup()
    {
        state = new Empty();
    }
}
```

This is the context class. This class contains an object of one of the children of the State class.

The getTemp() method, getMessage() method, and nextState() method both return what the State methods with similar names return.

The drankCup() method assigns the state object to an Empty object.

```
   Context drink;
    public Form1()
    {
        InitializeComponent();
        drink = new Context();
    }

    private void btnWait_Click(object sender, EventArgs e)
    {
        drink.nextState();
        lblLevel.Text = drink.getTemp();
```

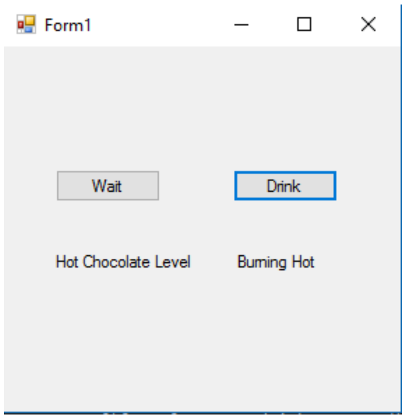
This is the Form class. It contains a Context object.

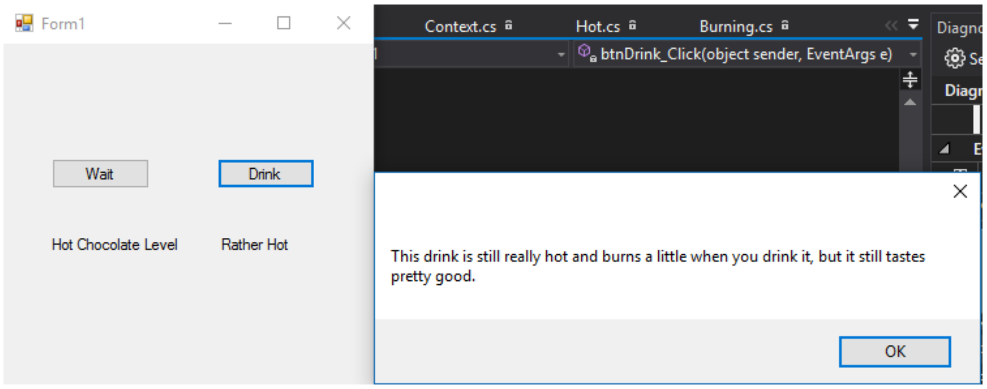
When the wait button is clicked, the Context.nextState() method is called, which changes the state, and the temperature is updated.

```
        }

        private void btnDrink_Click(object sender, EventArgs e)
        {
            MessageBox.Show(drink.getMessage());
            drink.drankCup();
            lblLevel.Text = drink.getTemp();
        }
    }
```

When the drink button is clicked, it shows the message in the message box, and changes the cup to an empty cup, and updates the label accordingly.

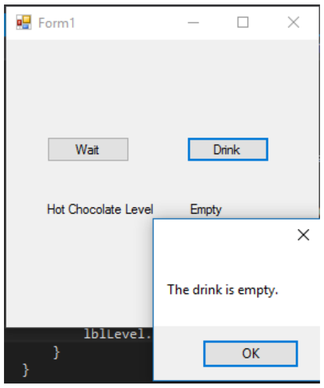




This drink is still really hot and burns a little when you drink it, but it still tastes pretty good.

Initial setup.

Waited until level was Hot, and Drink was pressed.



The drink is empty.

After drink was pressed first time, it changed to an empty cup, and nothing happens when drink is pressed again.

## Observations

When I initially started out on this, I was going to do an application that simulated jumping using a timer. I ran into problems with changing text in textboxes with the timer. I was listening to Christmas music and decided to change my application into one that demonstrates drinking hot chocolate instead because I wasn’t very high on the idea of the jumping application.