16. Шаблон Състояние (State)

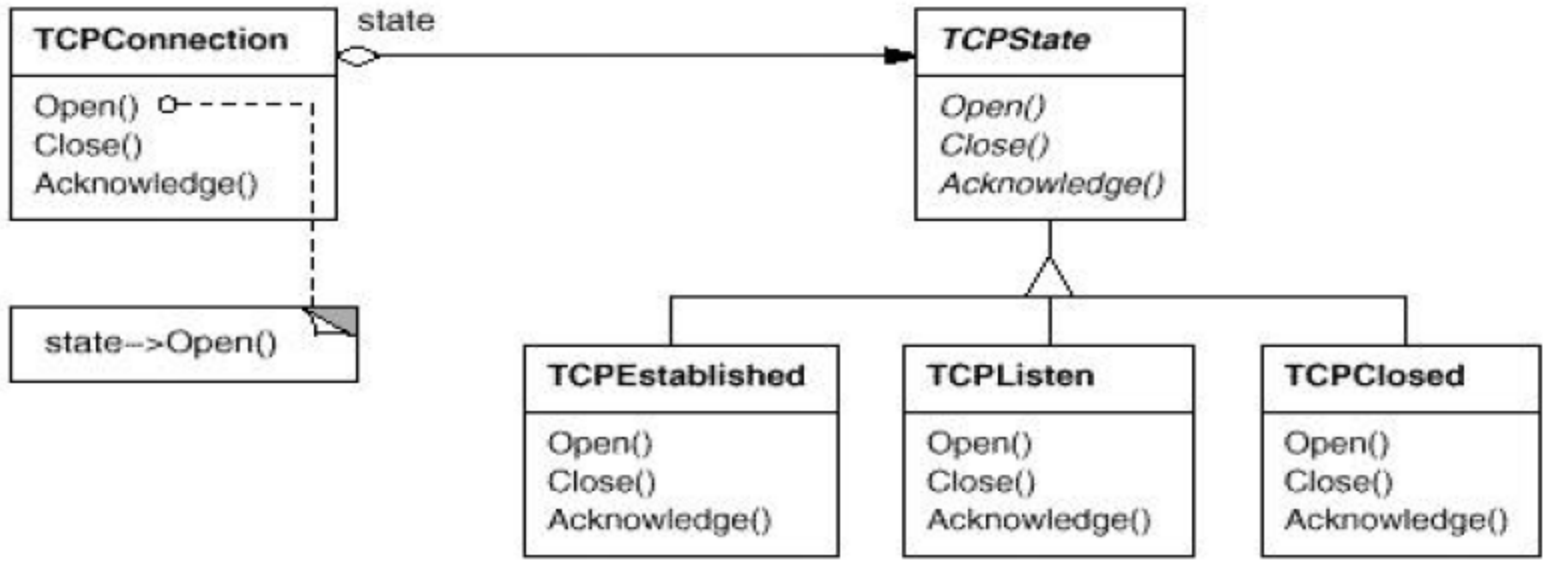
ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ

ДОЦ. Д-Р ЕМИЛ ДОЙЧЕВ

Общи сведения

- ✓ **Вид:** поведенчески за обекти
- ✓ **Цел:** Дава възможност на обект да променя поведението си когато се промени вътрешното му състояние. Ще изглежда така все едно обекта си е сменил класа.

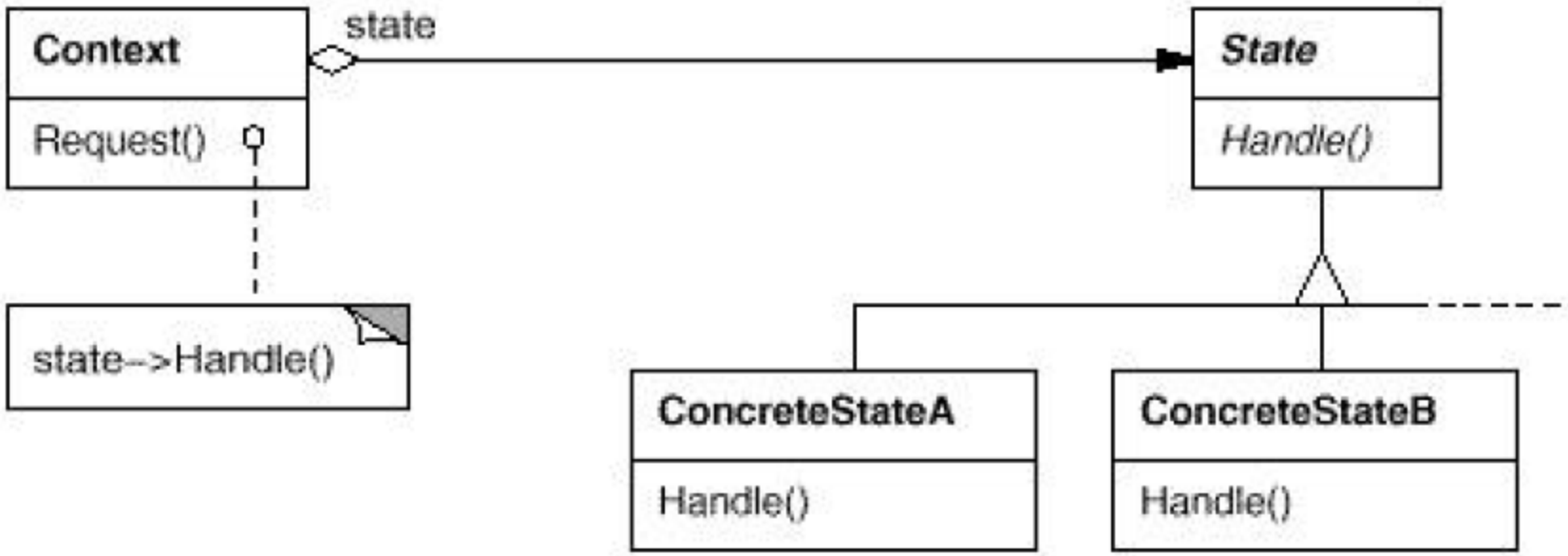
Мотивация


TCPConnection
Open()
Close()
Acknowledge()
state
state->Open()
TCPState
Open()
Close()
Acknowledge()
TCPEstablished
Open()
Close()
Acknowledge()
TCPListen
Open()
Close()
Acknowledge()
TCPClosed
Open()
Close()
Acknowledge()

Приложимост

- ✓ **Приложимост:** Шаблонът Състояние се използва в следните случаи:
  - Поведението на обекта зависи от вътрешното му състояние и трябва да може да променя поведението си в зависимост от състоянието си по време на изпълнение на програмата.
  - Методите съдържат големи и сложни условни конструкции, които зависят от състоянието на обекта. Шаблонът Състояние разполага всяко разклонение на условната конструкция в отделен клас.

Структура


Context
state
State
Request()
Handle()
state->Handle()
ConcreteStateA
Handle()
ConcreteStateB
Handle()

Следствия

- ✓ **Предимства**
  - Поставя цялото поведение, което е зависимо от състоянието, в един обект.
  - Позволява логиката по прехода от едно състояние в друго да бъде разположена в обекта-състояние, вместо монолитен if или switch израз.
  - Помага да се избегнат неконсистентните състояние тъй като промяната на състоянието се случва само в един обект-състояние, а не в набор от различни обекти или атрибути.
- ✓ **Недостатъци**
  - Увеличава се броя на обектите

Пример 1

- ✓ Разглеждаме клас, който има два метода – push и pull. Поведението на тези методи се променя в зависимост от състоянието на класа.
- ✓ За да изпращаме push и pull заявки към обекта ще използваме GUI подобно на показаното тук.

- ✓ Състоянието на обекта ще бъде показвано от цвета на прозореца.
- ✓ Състоянията са: черно, червено, синьо и зелено.

Пример 1

- ✓ Първо ще направим примера без да използваме шаблона Състояние

```
/**
 * Класа ContextNoSP има поведение, което зависи от състоянието му.
 * Методите push() и pull() извършват различни неща в зависимост от състоянието на обекта.
 * Този клас НЕ използва шаблона Състояние.
 */
public class ContextNoSP {
	// Състоянието!
	private Color state = null;

	// Създава нов ContextNoSP със зададеното състояние (color).
	public ContextNoSP(Color color) {
		state = color;
	}

	// Създава нов ContextNoSP със сътояние по подразбиране
	public ContextNoSP() {
		this(Color.red);
	}

	// Връща състоянието.
	public Color getState() {
		return state;
	}

	// Задава състояние.
	public void setState(Color state) {
		this.state = state;
	}
```

Пример 1

```
    /**
     * Методът push() извършва различни дейности в зависимост от състоянието на обекта.
     * Реално единственото нещо, което прави е смяна на състоянието.
     */
    public void push() {
        if (state == Color.red)
                state = Color.blue;
        else if (state == Color.green)
                state = Color.black;
        else if (state == Color.black)
                state = Color.red;
        else if (state == Color.blue)
                state = Color.green;
    }

    /**
     * Методът pull() извършва различни дейности в зависимост от състоянието на обекта.
     * Реално единственото нещо, което прави е смяна на състоянието.
     */
    public void pull() {
        if (state == Color.red)
                state = Color.green;
        else if (state == Color.green)
                state = Color.blue;
        else if (state == Color.black)
                state = Color.green;
        else if (state == Color.blue)
                state = Color.red;
    }
}
```

Пример 1

- ✓ Тестовата програма


```
/**
 * Тест програма за класа ContextNoSP, който НЕ използва шаблона състояние.
 */
public class TestNoSP extends Frame implements ActionListener {
      // GUI елементи.
      private Button pushButton = new Button("Push Operation");
      private Button pullButton = new Button("Pull Operation");
      private Button exitButton = new Button("Exit");
      private Canvas canvas = new Canvas();
      private Panel toolbox = new Panel();

      // Контекста
      private ContextNoSP context = null;

      public TestNoSP() {
            super("Без шаблона Състояние");
            context = new ContextNoSP();
            setupWindow();
      }
```

Пример 1

- ✓ Композиране на GUI


```
private void setupWindow() {
      setSize(500, 500);

      setLayout(new BorderLayout());
      add(canvas, BorderLayout.CENTER);

      toolbox.setLayout(new GridBagLayout());
      add(toolbox, BorderLayout.SOUTH);
      toolbox.add(pushButton);
      toolbox.add(pullButton);
      toolbox.add(exitButton);

      pushButton.addActionListener(this);
      pullButton.addActionListener(this);
      exitButton.addActionListener(this);
}
```

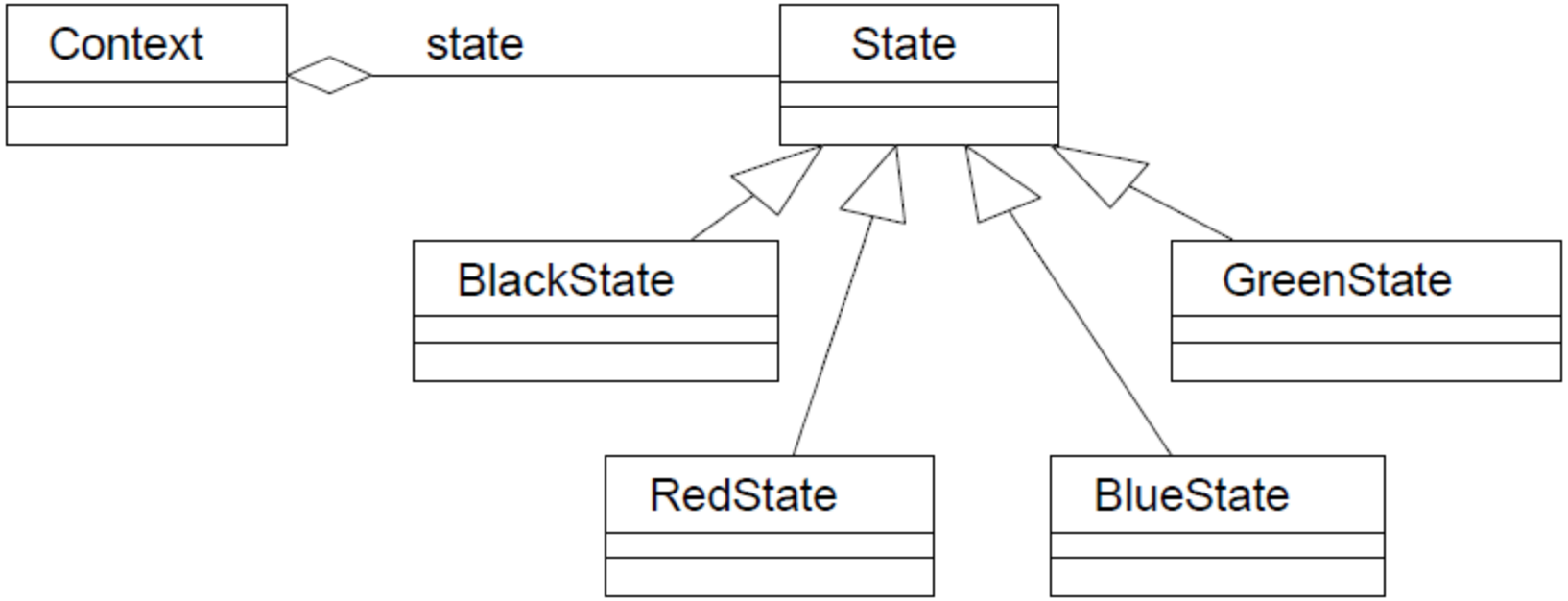
Пример 1


```
	// Обработка на събитията от бутоните.
	public void actionPerformed(ActionEvent event) {
		Object src = event.getSource();
		if (src == pushButton) {
			context.push();
			canvas.setBackground(context.getState());
		} else if (src == pullButton) {
			context.pull();
			canvas.setBackground(context.getState());
		} else if (src == exitButton) {
				System.exit(0);
		}
	}

	public static void main(String[] argv) {
		TestNoSP gui = new TestNoSP();
		gui.setVisible(true);
	}
}
```

Пример 1

- ✓ Сега да използваме шаблона Състояние



Пример 1

- ✓ Примера със използване на шаблона Състояние.
- ✓ Започваме с абстрактния клас, който представя състоянието на Context.

```
/**
 * Абстрактен клас, който дефинира интерфейса за поведението на
 * конкретно състояние на Context.
 */
public abstract class State {
      public abstract void handlePush(Context c);

      public abstract void handlePull(Context c);

      public abstract Color getColor();
}
```

Пример 1

- ✓ Примера със използване на шаблона Състояние

```
/**
 * Класът Context има поведение, което зависи от неговото състояние. Този клас
 * използва шаблона Състояние. Сега когато имаме pull() или push() заявки делегираме
 * поведението на обекта, който съдържа състоянието.
 */
public class Context {
	// Вътрешното състояние.
	private State state = null; // атрибут за състоянието

	// Създава нов Context със зададеното състояние.
	public Context(State state) {
		this.state = state;
	}

	// Създава нов Context със състояние по подразбиране.
	public Context() {
		this(new RedState());
	}

	public State getState() {
		return state;
	}

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

	...
```

Пример 1

```
        ...

        /**
         * Методът push() извършва различни действия в зависимосто от състоянието
         * на обекта. Като използваме шаблона Състояние делегираме поведението на този метод
         * на обекта, който съдържа самото състояние.
         */
        public void push() {
             state.handlePush(this);
        }

        /**
         * Методът pull() извършва различни действия в зависимосто от състоянието
         * на обекта. Като използваме шаблона Състояние делегираме поведението на този метод
         * на обекта, който съдържа самото състояние.
         */
        public void pull() {
             state.handlePull(this);
        }
}
```

Пример 1

- ✓ Имплементиране на класовете наследници на State съответно BlackState, GreenState, BlueState и RedState.

```
public class BlackState extends State {
    // Следващо състояние за Black състоянието:
    // При push(), преминава в "red"
    // При pull(), преминава в "green"

    @Override
    public void handlePush(Context c) {
        c.setState(new RedState());
    }

    @Override
    public void handlePull(Context c) {
        c.setState(new GreenState());
    }

    @Override
    public Color getColor() {
        return (Color.black);
    }
}
```

Край: Шаблон Състояние

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ