02

Salvando estados anteriores e o Memento

Imagine um sistema que gerencia contratos. Um contrato contém uma data, um cliente, e um tipo de contrato. O tipo pode ser "NOVO", "EM ANDAMENTO", "ACERTADO", "CONCLUIDO". Esses status mudam a medida que o contrato anda no processo.

Vamos representar essa classe. Repare no método `avanca()`, que muda o status do contrato de acordo com seu contrato atual.

```
public class Contrato {

    private Calendar data;
    private String cliente;
    private TipoContrato tipo;

    public Contrato(Calendar data, String cliente, TipoContrato tipo) {
        this.data = data;
        this.cliente = cliente;
        this.tipo = tipo;
    }

    // getters e setters

    public void avanca() {
        if(tipo == TipoContrato.NOVO) tipo = TipoContrato.EM_ANDAMENTO;
        else if(tipo == TipoContrato.EM_ANDAMENTO) tipo = TipoContrato.ACERTADO;
        else if(tipo == TipoContrato.ACERTADO) tipo = TipoContrato.CONCLUIDO;
    }

}
```

Agora, imagine que esses status de contrato mudem com frequência ao longo do tempo, e que precisamos de uma maneira de "voltar atrás" no estado de um contrato.

Como fazer isso? A primeira ideia é guardar uma lista dos vários contratos, cada um em um momento do tempo, com um estado diferente. Para tal, vamos criar a classe `Historico`, que guardará essa lista, e nos dará operações sobre ela, e a classe `Estado`, cuja ideia é encapsular um Contrato, e deixar o código mais explícito:

```
public class Estado {

    private Contrato contrato;

    public Estado(Contrato contrato) {
        this.contrato = contrato;
    }

    public Contrato getContrato() {
        return contrato;
    }

}
```

```
public class Historico {

    private List<Estado> estadosSalvos = new ArrayList<Estado>();

    public Estado pega(int index) {
        return estadosSalvos.get(index);
    }

    public void adiciona(Estado estado) {
        estadosSalvos.add(estado);
    }

}
```

Veja como estamos agora: temos a classe `Contrato`, que é uma classe de domínio, uma classe `Estado` que encapsula um Contrato em um determinado momento, e uma classe `Historico`, que guarda uma lista de estados.

Precisamos agora só "produzir esses estados" na hora certa. Para isso, vamos criar, na classe `Contrato`, um método que cria um `Estado`:

```
    public Estado salvaEstado() {
        return new Estado(new Contrato(data, cliente, tipo));
    }
```

Com isso, já conseguimos salvar estados nos momentos que queremos. Veja o exemplo abaixo:

```
public class Programa {

    public static void main(String[] args) {

        Historico historico = new Historico();

        Contrato contrato = new Contrato(Calendar.getInstance(), "Mauricio", TipoContrato.NOVO);
        historico.adiciona(contrato.salvaEstado());

        contrato.avanca();
        historico.adiciona(contrato.salvaEstado());

        contrato.avanca();
        historico.adiciona(contrato.salvaEstado());

        contrato.avanca();
        historico.adiciona(contrato.salvaEstado());
    }
}
```

Por fim, precisamos dar a opção para o usuário restaurar o estado de um contrato. Colocaremos esse método também dentro do Contrato. Veja só o método `restaura()`:

```
    public void restaura(Estado estado) {
        this.data = estado.getContrato().getData();
```

```
        this.cliente = estado.getContrato().getCliente();
        this.tipo = estado.getContrato().getTipo();
    }
```

Agora, conseguimos restaurar o estado de um contrato, pegando-o da lista de histórico. Veja o código que poderíamos colocar na classe `Programa` para isso funcionar:

```
contrato.restaura(historico.pega(1));
```

E pronto. Agora temos uma maneira eficiente de salvar estados de um objeto, e restaurá-los caso necessário. Sempre que temos um problema como esse, fazemos uso do **Memento**. O Memento é um padrão de projeto que nos ajuda a salvar e restaurar estados de objetos.