

INSTITUTO FEDERAL DE SANTA CATARINA - GASPAR
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ABDIEL DE ATHAYDE

Padrões de Projetos de Software
Atividade Prática

Padrão Singleton

O padrão Singleton garante que uma classe tenha apenas uma única instância em todo o programa, fornecendo um ponto global de acesso a essa instância. Na área de aviação, isso pode ser útil para gerenciar recursos que devem ser únicos, como o controle de tráfego aéreo.

```
# Uso do Singleton
controle1 = ControleTrafegoAereo()
controle2 = ControleTrafegoAereo()

print(controle1 is controle2) # True, pois ambas são a mesma instância

> class VooBuilder:
```

Padrão Builder

O padrão Builder é usado para construir objetos complexos de maneira passo a passo. Isso é útil na aviação para criar objetos complexos como um voo, que possui muitos atributos e possivelmente sub objetos, como tripulantes.

```
# Uso do Builder
builder = VooBuilder()
voo = (builder.set_numero_voo("AZ123")
      .set_companhia_aerea("Azul")
      .set_origem("GRU")
      .set_destino("JFK")
      .adicionar_comissario(ComissarioBordo("Maria Silva", "CB101"))
      .adicionar_comandante(Comandante("Carlos Souza", "CMD202", 15))
      .build())

print(voo)
```

Padrão Estrutural: Adapter

O padrão Adapter permite que objetos com interfaces incompatíveis trabalhem juntos. Na aviação, pode ser usado para integrar diferentes sistemas de gerenciamento de voos que utilizam interfaces distintas.

```
14
15 # Uso do Adapter
16 voo_adapter = AdapterVoo(voo)
17 sistema_externo = SistemaExterno()
18 sistema_externo.adicionar_voo_externo(voo_adapter.converter_para_externo())
19
20 print(sistema_externo.lista_voos)
21
```

Padrão Comportamental: Observer

O padrão Observer define uma dependência um-para-muitos entre objetos, onde um objeto notifica outros sobre mudanças em seu estado. Isso pode ser útil na aviação para notificar sistemas de controle ou usuários sobre mudanças em voos.

```
174
175 # Uso do Observer
176 voo_observable = VooObservable("AZ123")
177 passageiro1 = Passageiro("João")
178
179 voo_observable.adicionar_observador(passageiro1)
180 voo_observable.alterar_voo("GRU", "LAX") # Notifica os observadores sobre a mudança
181
```