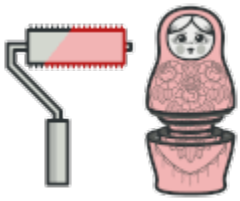
refactoring.guru

# Decorator em Go / Padrões de Projeto

~2 minutes

---



O **Decorator** é um padrão estrutural que permite adicionar novos comportamentos aos objetos dinamicamente, colocando-os dentro de objetos wrapper especiais.

Usando decoradores, você pode agrupar objetos inúmeras vezes, pois os objetos de destino e os decoradores seguem a mesma interface. O objeto resultante terá um comportamento de empilhamento de todos os wrappers.

## Exemplo conceitual

### pizza.go: Interface de componente

```
package main

type IPizza interface {
    getPrice() int
}
```

### veggieMania.go: Componente concreto

```
package main

type VeggieMania struct {
}

func (p *VeggieMania) getPrice() int {
    return 15
}
```

**tomatoTopping.go: Decorador concreto**

```
package main

type TomatoTopping struct {
    pizza IPizza
}

func (c *TomatoTopping) getPrice() int {
    pizzaPrice := c.pizza.getPrice()
    return pizzaPrice + 7
}
```

**cheeseTopping.go: Decorador concreto**

```
package main

type CheeseTopping struct {
    pizza IPizza
}
```

```
func (c *CheeseTopping) getPrice() int {
    pizzaPrice := c.pizza.getPrice()
    return pizzaPrice + 10
}
```

**main.go: Código cliente**

```
package main

import "fmt"

func main() {

    pizza := &VeggieMania{}

    //Add cheese topping
    pizzaWithCheese := &CheeseTopping{
        pizza: pizza,
    }

    //Add tomato topping
    pizzaWithCheeseAndTomato := &TomatoTopping{
        pizza: pizzaWithCheese,
    }

    fmt.Printf("Price of veggeMania with tomato and cheese topping
is %d\n", pizzaWithCheeseAndTomato.getPrice())
}
```

**output.txt: Resultados da execução**

Price of veggeMania with tomato and cheese topping is 32