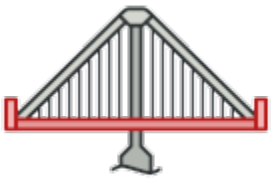
refactoring.guru

Bridge em Go / Padrões de Projeto

~4 minutes



O **Bridge** é um padrão de projeto estrutural que divide a lógica de negócio ou uma enorme classe em hierarquias de classe separadas que podem ser desenvolvidas independentemente.

Uma dessas hierarquias (geralmente chamada de Abstração) obterá uma referência a um objeto da segunda hierarquia (Implementação). A abstração poderá delegar algumas (às vezes, a maioria) de suas chamadas para o objeto de implementações. Como todas as implementações terão uma interface comum, elas seriam intercambiáveis dentro da abstração.

Exemplo conceitual

Digamos que você tenha dois tipos de computador: Mac e Windows. Além disso, dois tipos de impressoras: Epson e HP. Tanto os computadores quanto as impressoras precisam funcionar entre si em qualquer combinação. O cliente não quer se preocupar com os detalhes da conexão de impressoras a computadores.

Se introduzirmos novas impressoras, não queremos que nosso código cresça exponencialmente. Em vez de criar quatro structs para a combinação 2 * 2, criamos duas hierarquias:

- Hierarquia de abstração: serão nossos computadores
- Hierarquia de implementação: serão as nossas impressoras

Essas duas hierarquias se comunicam por meio de uma Bridge, onde a Abstração (computador) contém uma referência à Implementação (impressora). Tanto a abstração quanto a implementação podem ser desenvolvidas independentemente, sem afetar uma à outra.

computer.go: Abstração

```
package main

type Computer interface {
    Print()
    SetPrinter(Printer)
}
```

mac.go: Abstração refinada

```
package main

import "fmt"

type Mac struct {
    printer Printer
}
```

```
func (m *Mac) Print() {
    fmt.Println("Print request for mac")
    m.printer.PrintFile()
}

func (m *Mac) SetPrinter(p Printer) {
    m.printer = p
}
```

windows.go: Abstração refinada

```
package main

import "fmt"

type Windows struct {
    printer Printer
}

func (w *Windows) Print() {
    fmt.Println("Print request for windows")
    w.printer.PrintFile()
}

func (w *Windows) SetPrinter(p Printer) {
    w.printer = p
}
```

printer.go: Implementação

```
package main

type Printer interface {
    PrintFile()
}
```

epson.go: Implementação concreta

```
package main

import "fmt"

type Epson struct {
}

func (p *Epson) PrintFile() {
    fmt.Println("Printing by a EPSON Printer")
}
```

hp.go: Implementação concreta

```
package main

import "fmt"

type Hp struct {
}

func (p *Hp) PrintFile() {
    fmt.Println("Printing by a HP Printer")
```

```
}
```

main.go: Código cliente

```
package main

import "fmt"

func main() {

    hpPrinter := &Hp{}
    epsonPrinter := &Epson{}

    macComputer := &Mac{}

    macComputer.SetPrinter(hpPrinter)
    macComputer.Print()
    fmt.Println()

    macComputer.SetPrinter(epsonPrinter)
    macComputer.Print()
    fmt.Println()

    winComputer := &Windows{}

    winComputer.SetPrinter(hpPrinter)
    winComputer.Print()
    fmt.Println()

    winComputer.SetPrinter(epsonPrinter)
```

```
    winComputer.Print()
    fmt.Println()
}
```

output.txt: Resultados da execução

```
Print request for mac
Printing by a HP Printer

Print request for mac
Printing by a EPSON Printer

Print request for windows
Printing by a HP Printer

Print request for windows
```

Bridge em outras linguagens