05

Para saber mais

Passando um valor ou uma cópia

Métodos são definidos de maneira parecida com funções, mas de uma maneira diferente. Existe um `(p *Pessoa)` que se refere a um ponteiro para a instância criada da estrutura, conforme o exemplo abaixo:

```
package main

import (
    "fmt"
)

type Pessoa struct {
    nome, sobrenome string
}

func (p *Pessoa) ExibirNomeCompleto() string {
    nomeCompleto := p.nome + " " + p.sobrenome
    return nomeCompleto
}

func main() {
    p1 := Pessoa{"Guilherme", "Lima"}
    fmt.Println(p1.ExibirNomeCompleto())
}
```

Neste link, você pode executar o código acima (https://play.golang.org/p/m1yagmicHND)

Ao executar este código, temos a saída esperada:

```
Guilherme Lima

Program exited.
```

Nesse caso, passamos para o método o valor encontrado neste ponteiro através do `(p *Pessoa)`.

Passando uma cópia

Também é possível passar um valor removendo a assinatura do ponteiro `(p *Pessoa)` para `(p Pessoa)`.

Nesse caso, uma **cópia** do valor de Pessoa é passada para a função, sem alterar o valor do ponteiro. Portanto, precisamos ficar atento, já que qualquer alteração que você faça em `p` se passar por valor não será refletida na fonte `p`.

Observe este exemplo:

```
package main
```

```
import (
    "fmt"
)

type Pessoa struct {
    nome, sobrenome string
}

func (p Pessoa) ExibirNomeCompleto() string {
    p.sobrenome = "Silva"
    nomeCompleto := p.nome + " " + p.sobrenome
    return nomeCompleto
}

func main() {
    p1 := Pessoa{"Guilherme", "Lima"}

    fmt.Println(p1.ExibirNomeCompleto())
    fmt.Println(p1.nome, p1.sobrenome)
}
```

Neste link, você pode executar o código acima (https://play.golang.org/p/GVfCcf3ag53)

Nossa saída será:

Observe que alteramos o sobrenome de `p` no método `ExibirNomeCompleto`, mas não foi alterado o valor armazenado no ponteiro. Sendo assim, quando não precisamos alterar o conteúdo de um ponteiro, podemos passar apenas uma cópia.