Go
6-structure,méthode et interface

J. Vlasak

Go 6- structure,méthode et interface

But du cours

- Savoir manipuler les structure, méthode et interface

Go 6 - structure,méthode et interface

Structure et méthode

- Revenons sur la structure Mouton du TP-5

```
type Sheep struct {
    nom string
    age int
    poids float32
}
```

- Et de la fonction d'affichage

```
func  PrintSheep (s *Sheep) {
    fmt.Println(s.nom, s.age, s.poids)
}
```

```
sheep := Sheep{}
PrintSheep(&sheep)
```

- Go permet de définir des méthodes sur les types définies par une structure

```
func (s *Sheep) PrintSheep () {
    fmt.Println(s.nom, s.age, s.poids)
}
```

```
sheep := Sheep{}
sheep.PrintSheep()
```

Go 6- structure,méthode et interface

Structure et méthode

- Pas de notion d'héritage, utilise la composition

```
type Animal struct {
    nom string
    age int
}

...
type Sheep struct {
    animal Animal
    poids  float32
}
```

```
func PrintSheep(s *Sheep) {
    fmt.Println(s.animal.nom, s.animal.age, s.poids)
}
```

- Notion de type embarqué

```
type Sheep struct {
    Animal
    poids float32
}
```

```
func PrintSheep(s *Sheep) {
    fmt.Println(s.Animal.nom, s.Animal.age, s.poids)
}
```

Structure et méthode

- Un nom unique pour une méthode, pas de signature différente

```
func (s *Sheep) PrintSheep() {
    fmt.Println(s.nom, s.age, s.poids)
}

func (s *Sheep) PrintSheep(commentaire string) {
    fmt.Println(s.nom, s.age, s.poids)
}
```

- Pas de getter et setter nécessaire.
- Une méthode est une fonction

```
sheep := Sheep{}
ptrFonction := sheep.PrintSheep
ptrFonction()
```

Go 6- structure,méthode et interface

Structure et méthode : pointeur

- Peut utiliser un pointeur ou un passage par valeur

Valeur immuable

```go
func (s Sheep) PrintSheep() {
    fmt.Println(s.nom, s.age, s.poids)
}

func (s *Sheep) Grossi(v float32) {
    s.poids += v
}
```

Modification de Sheep

Go 6- structure,méthode et interface

Interface

- Le terme interface en Go est utilisé de plusieurs façons toutes liées à la notion d’un contrat de fonctionnalité.
- Les types interfaces sont des types définis, caractérisés par leur jeu de méthodes, également dénommé interface
- la satisfaction des interfaces est implicite : cela signifie que tout type comportant l’ensemble des méthodes d’une interface est reconnu comme implémentant cette interface. Pas de mot clef spécifique.

Go 6- structure,méthode et interface

Interface : un exemple

- Un exemple avec le calcul d'une aire

```
type Forme interface {
    aire() float64
}
```


```
You, 1 second ago | 1 author (You)
type Cercle struct {
    Rayon float64
}

func (c Cercle) aire() float64 {
    return 3.14 * c.Rayon * c.Rayon
}
```

```
func afficherAire(f Forme) {
    aire := f.aire()
    fmt.Printf("Aire de la forme : %f\n", aire)
}
```

```
func main() {
    cercle := Cercle{Rayon: 5}
    rectangle := Rectangle{Largeur: 4, Hauteur: 6}

    afficherAire(cercle)
    afficherAire(rectangle)
```

Go 6- structure,méthode et interface

Interface : cas particulier

- Possibilité de stoker n’import quoi dans une variable

```
sheep := Sheep{}
  You, 15 minutes ago • v
var quelquechose interface{}
quelquechose = sheep
```

- Encapsule la données dans une structure non manipulable directement (type + donnée)

```
quelquechose: interface {}(main.Sheep) {Animal: main.Animal {nom: "", age: 0}, poids: 10}
  data: main.Sheep {Animal: main.Animal {nom: "", age: 0}, poids: 10}
    Animal: main.Animal {nom: "", age: 0}
    poids: 10
```

- Utile dans le cas du traitement de données non structurées (analyse de ficher yaml, json, ...)

Interface : cas particulier

- Si on veux modifier la valeur de sheet, obligé de tester le type. Ce qui n'est pas le cas en lecture.

```
var quelquechose interface{}
quelquechose = sheep

vP, ok := quelquechose.(Personne)

fmt.Println(quelquechose.(Sheep).age)
quelquechose.(Sheep).age = 11
  You, yesterday • v
switch v := quelquechose.(type) {
case Sheep:
    fmt.Println(v.age)
    v.age = 10

}
```

Évite un crash du programme

Ne compile pas

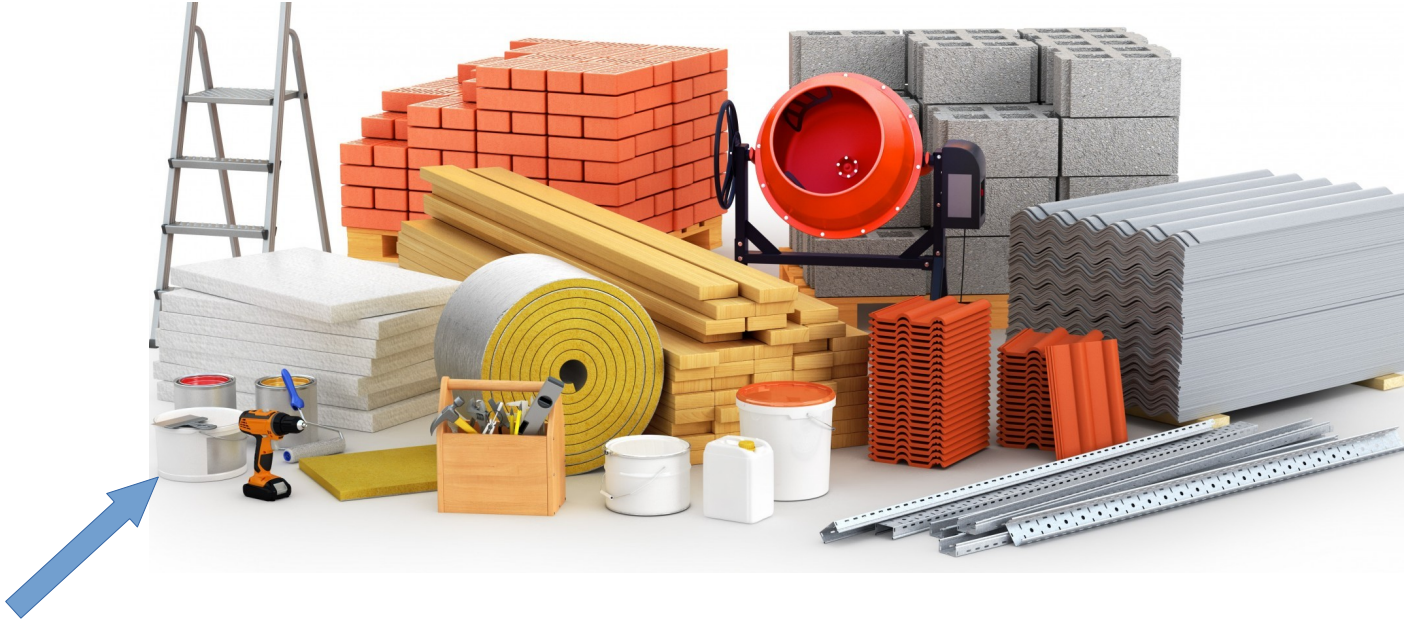
Interface : cas des test unitaires

- A completer ; Permet de faire des mock d'object dans des tests.

Évite un crash du programme

Ne compile pas

Go 5-Type composé



Taille définie à la compilation