**ЛАБОРАТОРНА РОБОТА №5**
**ООП. Структури та інтерфейси**

**Мета роботи:** засвоїти принципи проектування та оголошення структурта інтерфейсів; вивчити особливості реалізації структур та їх методів.

**Хід роботи:**

**Варіант : непарний**

**Завдання 1:** Оголосити структуру Product, яка представляє інформацію про один товар, який зберігається на складі. Має наступні поля:

- Name – назва товару;
- Price - вартість одиниці товару;
- Cost – грошова одиниця, у якій вимірюється вартість;
- Quantity – кількість наявних товарів на складі;
- Producer – назва компанії-виробника;
- Weight – вага одиниці товару.

**Результат роботи:**

```
type Product struct {
    name     string
    price    float64
    cost     Currency
    quantity int64
    producer string
    weight   float64
}
```

Оголосив структуру Currency, яка містить наступні поля:

- Name – назва валюти;
- ExRate – курс (дробове число - кількість гривень та копійок, що дають за одну одиницю валюти).

**Результат роботи:**

```
type Currency struct {
    name   string
    exRate float64
}
```

**Завдання 2:** Для кожної структури реалізував конструктори:

```go
func ProductConstructor(name string, price float64, cost Currency,
    quantity int64, producer string, weight float64) *Product {
    p := new(Product)

    p.name = name
    p.price = price
    p.cost = cost
    p.quantity = quantity
    p.producer = producer
    p.weight = weight

    return p
}

func ProductConstructorWithoutCost(name string, price float64,
    quantity int64, producer string, weight float64) *Product {
    p := new(Product)

    p.name = name
    p.price = price
    p.cost = Currency{"USD", 36.8}
    p.quantity = quantity
    p.producer = producer
    p.weight = weight

    return p
}

func ProductDefaultConstructor() *Product {
    p := new(Product)

    p.name = "Tab M10 (3rd Gen) 4/64 Wi-Fi Storm Grey"
    p.price = 230.7
    p.cost = Currency{"USD", 36.8}
    p.quantity = 54
    p.producer = "Lenovo"
    p.weight = 423.7

    return p
}
```

**Завдання 3:** Реалізував set- та get- методи для кожного поля Product.

```go
func (p *Product) SetName(name string) {
    p.name = name
}

func (p *Product) SetPrice(price float64) {
    p.price = price
}

func (p *Product) SetCost(cost Currency) {
    p.cost = cost
}

func (p *Product) SetWeight(weight float64) {
    p.weight = weight
}

func (p *Product) SetProducer(producer string) {
    p.producer = producer
}

func (p *Product) SetQuantity(quantity int64) {
    p.quantity = quantity
}
```

```go
func (p *Product) GetName() string {
    return p.name
}

func (p *Product) GetPrice() float64 {
    return p.price
}

func (p *Product) GetCost() Currency {
    return p.cost
}

func (p *Product) GetQuantity() int64 {
    return p.quantity
}

func (p *Product) GetWeight() float64 {
    return p.weight
}

func (p *Product) GetProducer() string {
    return p.producer
}
```

**Завдання 4:** У структурі Product створив методи:

- *GetPriceIn()*, який повертає ціну одиниці товару в гривнях;
- *GetTotalPrice()*, що повертає загальну вартість усіх наявних на складі товарів даного виду;
- *GetTotalWeight()*, який повертає загальну вагу усіх товарів на складі даного виду.

У структурі *Currency* методи обрати самостійно.

```go
func (p *Product) GetInfo() string {
    priceStr := fmt.Sprintf("%.2f", p.GetPrice())
    costInfo := p.cost.GetInfo()
    quantityStr := fmt.Sprintf("%d", p.GetQuantity())
    weightStr := fmt.Sprintf("%.2f", p.GetWeight())

    result := "Name: " + p.GetName() + "\n"
    result += "Price: " + priceStr + "\n"
    result += "Cost: " + costInfo + "\n"
    result += "Quantity: " + quantityStr + "\n"
    result += "Producer: " + p.GetProducer() + "\n"
    result += "Weight: " + weightStr + "\n"

    return result
}

func (p *Product) GetPriceIn() float64 {
    cost := p.GetCost()
    exRate := cost.GetExRate()

    return exRate * p.GetPrice()
}

func (p *Product) GetTotalPrice(isUah bool) float64 {
    if isUah {
        return p.GetPriceIn() * float64(p.GetQuantity())
    }

    return p.GetPrice() * float64(p.GetQuantity())
}

func (p *Product) GetTotalWeight() float64 {
    return p.GetWeight() * float64(p.GetQuantity())
}
```

**Завдання 5:** Реалізував ф-ї:

- *ReadProductsArray()* – читає з клавіатури дані і повертає множину об'єктів типу *Product* (*n* об'єктів з різною валютою);
- *PrintProduct()* – приймає тип *Product* і виводить його на екран;
- *PrintProducts()* – приймає множину типу *Product* і виводить його на екран;
- *GetProductsInfo()* – приймає множину типу *Product* і повертає через вихідні параметри найдешевший та найдорожчий товар.

**Результати роботи програми :**

```
Введіть обраний варіант: 3

Виведення інформації про продукти...

Продукт # 1
Name: Tab M10 (3rd Gen) 4/64 Wi-Fi Storm Grey
Price: 230.70
Cost: 1 USD = 36.8 UAH
Quantity: 54
Producer: Lenovo
Weight: 423.70
```

```
Продукт # 2
Name: asus
Price: 40.00
Cost: 1 USD = 36.84 UAH
Quantity: 10
Producer: Asus
Weight: 3.00


Продукт # 3
Name: Tab M10 (3rd Gen) 4/64 Wi-Fi Storm Grey
Price: 230.70
Cost: 1 USD = 36.8 UAH
Quantity: 54
Producer: Lenovo
Weight: 423.70
```

```
Визначення мінімального і максимального продуктів...
Найдешевший продукт: asus 1473.60 UAH
Найдорожчий продукт: Tab M10 (3rd Gen) 4/64 Wi-Fi Storm Grey 8489.76 UAH
PS C:\Users\raf1x\Desktop\GO\Lab5>
```

## Контрольні питання

### 1. Що таке структура і для чого призначена?

*Структури – набір різних типів даних, що зберігаються якєдинецілетапередбачають доступ до окремих полів структури. Для доступу до полів структури використовується точкова нотація. Структури доцільно використовувати там, де необхідно об'єднатидані,що відносяться до одного об'єкту*

Загальний синтаксис визначення шаблону структури має вигляд:

```
var ім'я struct {
   ім'я_змінної1 тип1
   ім'я_ змінної2 тип2
   // інші члени структури
}
```

### 2. Як створюються користувацький тип?

*Роль класів виконують користувацькі типи (в основному структури). Структури можуть мати методи. Метод – це особливого роду функція, яка викликається відноснозначеннякористувацького типу (передається методу, що викликається). Значенняпередається за вказівником або за*

*значенням, в залежності відтого,якоголошений метод. Синтаксис методу аналогічний функції, тільки перед ім'ям методу вказується користувацький тип до якого він відноситься.*

Наприклад:

```
type Count struct { c uint64 }

func (this *Count) Inc() { this.c++ }
func (this *Count) Dec() { this.c-- }
func (this *Count) Zero() { this.c = 0 }
func (this Count) IsZero() bool { return this.c == 0 }
```

**3. Що таке ООП?**

*Об'є́ктоорієнто́ване програмува́ння — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція Однією з переваг ООП є краща модульність програмного забезпечення*

**4. Для чого використовується вбудова користувацьких типів?**

*Складові типи на основі структур можуть включати одинабобільшетипів у вигляді вбудованих полів. Головна зручність такого підходуполягаєвможливості викликати методи вбудованого типу щодозначеннякористувацького типу, так якби вони були власними методами цьоготипу. Методи вбудованих полів можна перевизначать, просто створюючи, дляструктури в яку вбудовують, нові методи з тими ж іменами, щоі методивбудованого поля*

**5. Що таке інтерфейси і для чого вони призначені?**

*Інтерфейс – це тип, який визначає сигнатури одного або більшеметодів.Інтерфейси є повністю абстрактними, тому немає ніякої можливостістворювати їх екземпляри. Однак є можливість створювати змінні зтипамиінтерфейсів, яким потім можна присвоювати значення будь-якогоконкретноготипу, що володіє методами інтерфейсу.*

**Висновок:** засвоїв принципи проектування та оголошення структура інтерфейсів; вивчив особливості реалізації структур та їх методів.