# Object template

## 簡述

使用類別 (class) 或結構 (struct) 化為特定型別與邏輯定義。物件以樣板建構，樣板使用預設值或提供初始化設定。

## 優點

封裝物件，防止耦合。

## 實作

### 物件建立

```swift
class Product {

    // MARK: - 欄位定義
    var name:String
    var description:String
    var price:Double
    var stockQuantity:Int = 0

    // 庫存
    var stock:Int{
        get{
            return stockQuantity
        }
        set{
            // 取較大值
            stockQuantity = max(0, newValue)
        }
    }

    // MARK: - 初始化設定
    init(name:String, description:String, price:Double, stock:Int){
        self.name = name
        self.description = description
        self.price = price
        self.stock = stock
    }

    // 計算稅金
    func calculateTax(rate: Double) -> Double{
        return self.price * rate
    }

    // 庫存總值
    var stockValue: Double{
        get{
            return self.price * Double(self.stock)
        }
    }
}
```

### 使用物件

```swift
var product = Product(name: "coffee", description: "latté", price: 50, stock: 10)

var products = [Product(name: "Kayak", description: "A boat for one person", price: 50, stock: 10),
                Product(name: "Lifejacket", description: "Protective and fashionable", price: 10, stock: 20),
                Product(name: "Soccer Ball", description: "FIFA-approved size and weight", price: 5, stock: 30)]

func calculateStockValue(productsArray:[Product]) -> Double {

    // reduce : 用於將一個集合中的元素依照指定的規則合併為單一值
    return productsArray.reduce(0, {(total,product) -> Double in
        return total + product.stockValue
    })
}

print("Sales tax for Kayak: $\(products[0].calculateTax(rate: 0.1))")
print("Total value of stock: $\(calculateStockValue(productsArray: products))")
```