Class và Struct

Class

- **Swift** là ngôn ngữ kế thừa ngôn ngữ **C** và **Objective-C**, nó vừa là một ngôn ngữ hướng thủ tục, vừa là một ngôn ngữ hướng đối tượng.
- **Class** chính là khái niệm của các ngôn ngữ lập trình hướng đối tượng
- **Class** có các thuộc tính và phương thức, về bản chất phương thức (method) được hiểu là một hàm của lớp.
- Từ một lớp, có thể tạo ra các đối tượng.

```
class <tên Class>: <Kế thừa Class cha, nếu có> {
    // Khai báo thuộc tính
    // Khởi tạo phương thức
}
```

Khai báo class với khởi tạo không tham số

- **init** là trình khởi tạo - một phương thức (**method**) mà **class** sẽ gọi đến khi chúng ta tạo một đối tượng **Person**.

```
class Rectangle1  {

    // thuộc tính chiều rộng, chiều cao
    var width: Int = 5
    var height: Int = 10

    // Constructor (khởi tạo) mặc định (Không tham số)
    // (Được sử dụng để tạo ra đối tượng)
    init()  {
    }

    // Phương thức dùng để tính diện tích hình chữ nhật.
    func getArea()  -> Int  {

        var area = self.width * self.height
        return area
    }
}
```

Khởi tạo đối tượng Rectangle1

```
// Tạo một đối tượng Rectangle1
// thông qua Constructor mặc định: init()
var rec1  = Rectangle1()

// In ra width, height.
print("rec1.width = \(rec1.width)")
print("rec1.height = \(rec1.height)")

// Gọi phương thức để tính diện tích.
 print("area = \(rec1.getArea())")
```

Khai báo class với khởi tạo có tham số

```
class Rectangle2 {

    // thuộc tính chiều rộng, chiều cao
    var width: Int
    var height: Int

    // Một Constructor có 2 tham số.
    // (Được sử dụng để tạo ra đối tượng)
    // self.width trỏ tới thuộc tính (property) width của class.
    init (width: Int, height: Int)  {
        self.width = width
        self.height = height
    }

    // Phương thức dùng để tính diện tích hình chữ nhật.
    func getArea()  -> Int  {

        var area = self.width * self.height
        return area
    }
}
```

Khởi tạo đối tượng Rectangle2

```
// Tạo đối tượng Rectangle2
// thông qua Constructor có 2 tham số: init(Int,Int)
var rec2 = Rectangle2(width: 10, height: 15)

// In ra width, height.
print("rec2.width = \(rec2.width)")
print("rec2.height = \(rec2.height)")

// Gọi phương thức để tính diện tích.
print("area = \(rec2.getArea())")
```

Deinitializer

- Deinitializer cho phép hàm khởi tạo của một class giải phóng bất cứ tài nguyên nào mà nó đã gán.
- Một hàm deinitializer được gọi ngay trước khi instance của một class được giải phóng
- Hàm deinit chỉ có sẵn trong class

```
class D {
    deinit {
        print("Deinit ")
    }
}

var d: D? = D()
d = nil
```

Struct là gì

- Trong **Swift**, **Struct** (cấu trúc) là một kiểu giá trị đặc biệt, nó tạo ra một biến để lưu trữ nhiều giá trị riêng lẻ, mà các các trị này có liên quan tới nhau
- Ví dụ về một nhân viên bao gồm: mã nhân viên, tên nhân viên, chức vụ
- Chúng ta hoàn toàn có thể tạo ra 3 biến để lưu các trường giá trị của một nhân viên. Tuy nhiên chúng ta nên tạo ra một Struct để gộp tất cả thông tin vào một biến duy nhất.

```
struct <tên Struct>: <kế thừa 1 hoặc nhiều giao thức> {
    // Khai báo thuộc tính
    // Khai báo phương thức
}
```

Khai báo struct

```
struct Employee {

    // khai báo thuộc tính
    var empNumber: String
    var empName: String
    var position: String

    // Constructor (khởi tạo)
    init(empNumber:String, empName:String, position:String) {
        self.empNumber = empNumber;
        self.empName = empName;
        self.position = position;
    }
}
```

Khởi tạo một đối tượng Employee

```
func test_EmployeeStruct()     {

    // Tạo một biến kiểu struct Employee.
    let john = Employee(empNumber:"E01",empName: "John", position:
"CLERK")

    print("Emp Number: " + john.empNumber)
    print("Emp Name: " + john.empName)
    print("Emp Position: " + john.position)

}
```

Phương thức khởi tạo của Struct

- Struct có thể có các Constructor (Phương thức khởi tạo), nhưng không có Destructor (Phương thức huỷ đối tượng)
- Bạn có thể không viết, viết một hoặc nhiều constructor cho Struct
- Trong khởi tạo, chúng ta phải gán tất cả các trường chưa có giá trị

```
// struct không có hàm khởi tạo
struct Employee {

    var empNumber:String
    var empName:String
    var position:String

}
```

Phương thức và thuộc tính của Struct

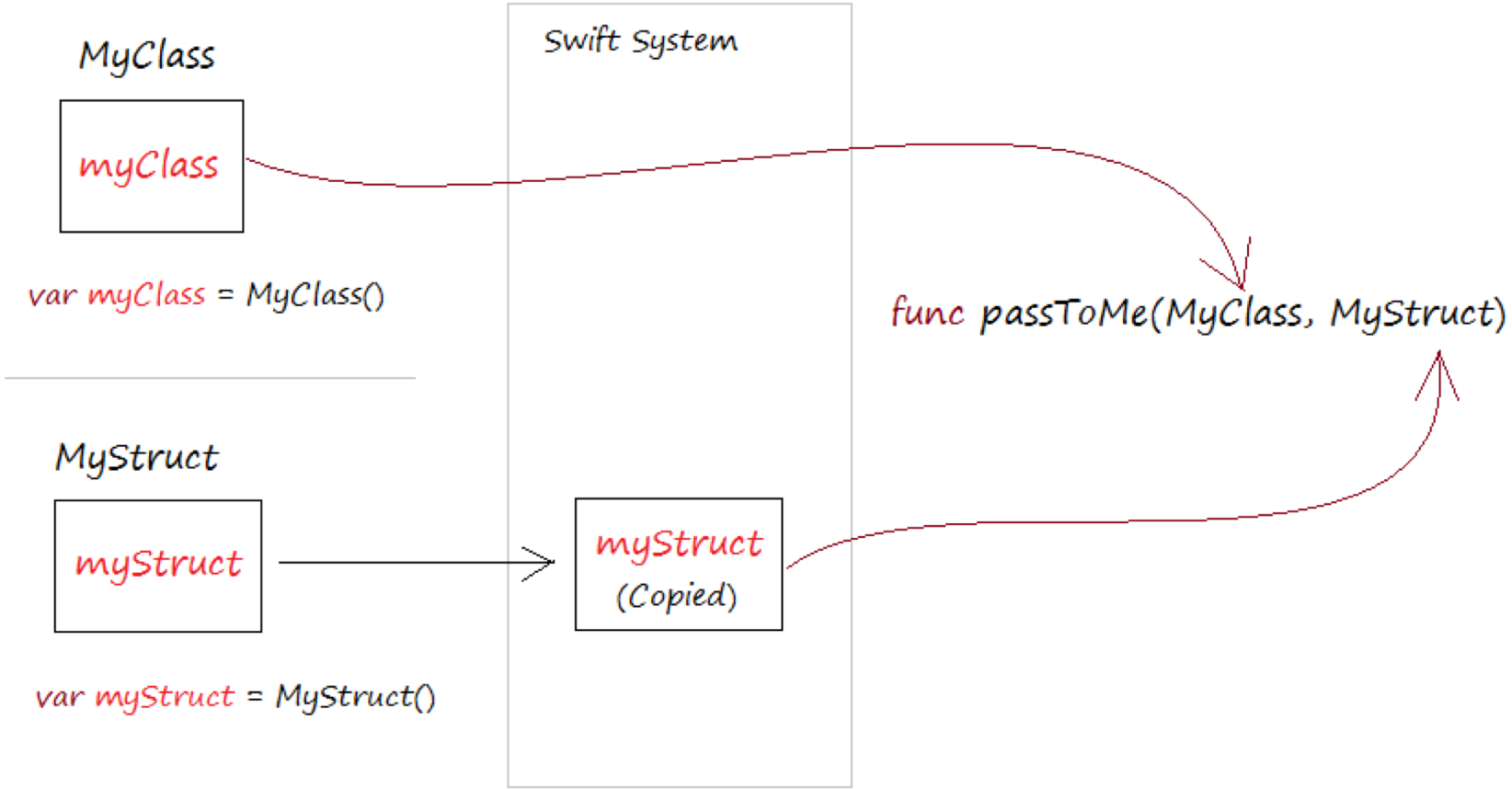
- Struct có thể có các phương thức và thuộc tính:

```
struct Book  {

    // thuộc tính
    var title: String
    var author: String

    // Khởi tạo
    init(  title:String, author:String) {
        self.title = title
        self.author = author
    }

    // Phương thức
    func getInfo() -> String {
        return "Book Title: " + self.title + ", Author: " + self.author
    }
}
```

Struct và Class

- Struct thường được sử dụng để tạo ra model (đối tượng) để lưu trữ giá trị, Class thì được sử dụng đa dạng hơn
- Struct không cho phép kế thừa từ một class hay một struct khác
- Nhưng struct cho phép kế thừa từ 1 hoặc nhiều Protocol
- Struct là kiểu tham số, Class là kiểu tham trị
- Nếu **struct** xuất hiện như một tham số trong một hàm (hoặc phương thức), nó được truyền (pass) dưới dạng giá trị. Trong khi đó nếu đối tượng của **class** xuất hiện như là một tham số trong một hàm (hoặc phương thức) nó được truyền (pass) như một tham chiếu (reference).

Struct – Class: Tham chiếu và tham trị



Ví dụ về tham chiếu và tham trị

```
class Person {
    var name: String
    var age: Int

    init(name: String, age: Int) {
        self.name = name
        self.age = age
    }
}


let person1 = Person(name: "A", age: 13)
var person2 = person1
person2.age = 15


print(person1.age) // kq: 15
print(person2.age) // kq: 15
```

```
struct Person {
    var name: String
    var age: Int

    init(name: String, age: Int) {
        self.name = name
        self.age = age
    }
}


let person1 = Person(name: "A", age: 13)
var person2 = person1
person2.age = 15


print(person1.age) // kq: 13
print(person2.age) // kq: 15
```

Chọn sử dụng class hay struct

- Chúng ta đã biết class có instances được truyền bởi tham chiếu, còn struct được truyền bởi giá trị. Vậy nên tuỳ vào cấu trúc dữ liệu hay chức năng mà chúng ta quyết định sử dung class hay struct:
- **Chúng ta xem xét việc tạo struct khi:**

1. Cấu trúc dữ liệu đơn giản, có ít thuộc tính
2. Những dữ liệu được đóng gói sẽ được copy hơn là tham chiếu khi bạn gán hay truyền instance của struct đó.
3. Những thuộc tính được lưu trữ bởi struct thì bản thân nó là kiểu giá trị.
4. Struct không cần thừa kế thuộc tính hay hành vi từ bất kì kiểu khác.