Swift Memory Management

Pointer là gì?

- Con trỏ được dùng để trỏ tới tới địa chỉ trên bộ nhớ nơi được dùng để lưu trữ giá trị của biến

Stack và Heap

- Stack: Là nơi lưu trữ các biến cục bộ, và tự động được giải phóng khi kết thúc một scope. Kích thước vùng nhớ stack được fix cố định. Chúng ta không thể tăng hoặc giảm kích thước vùng nhớ stack

- Heap: Được dùng cho cấp phát bộ nhớ động và tồn tại đến khi dev giair phóng vùng nhớ đó. Hệ điều hành sẽ có cơ chế tăng kích thước vùng nhớ heap.

Value Types và Reference types

- Value type(tham trị) : Int, Double, Struct...
- Reference type (tham chiếu) : Closure, Class

```
var number1: Int = 0
var number2 = number1


class User{
    var age = 20
    init(age: Int){
        self.age = age
    }
}
var user1: User = User(age: 21)
```

Reference counting

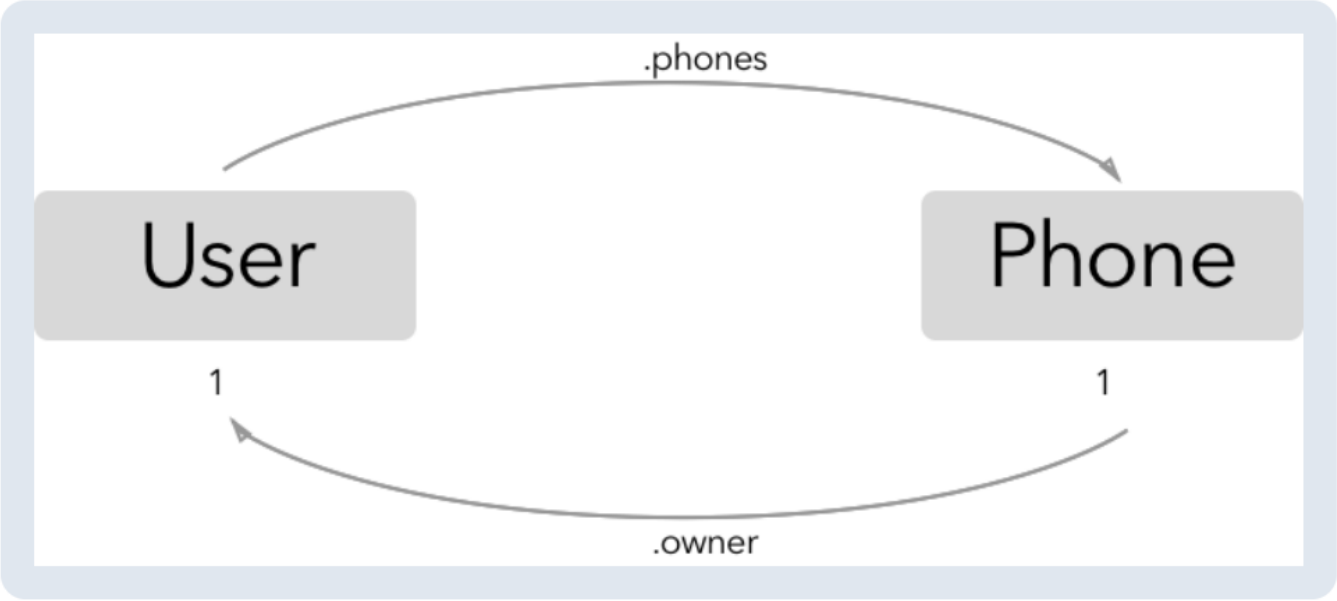
- Retain count là gì?
- Các loại reference trong Swift
  - Strong reference
  - Weak reference
  - Unowned Reference

Automatic Reference Counting (ARC)

- Cơ chế hoạt động ARC và non-ARC
- Cơ chế quản lý bộ nhớ ngôn ngữ khác

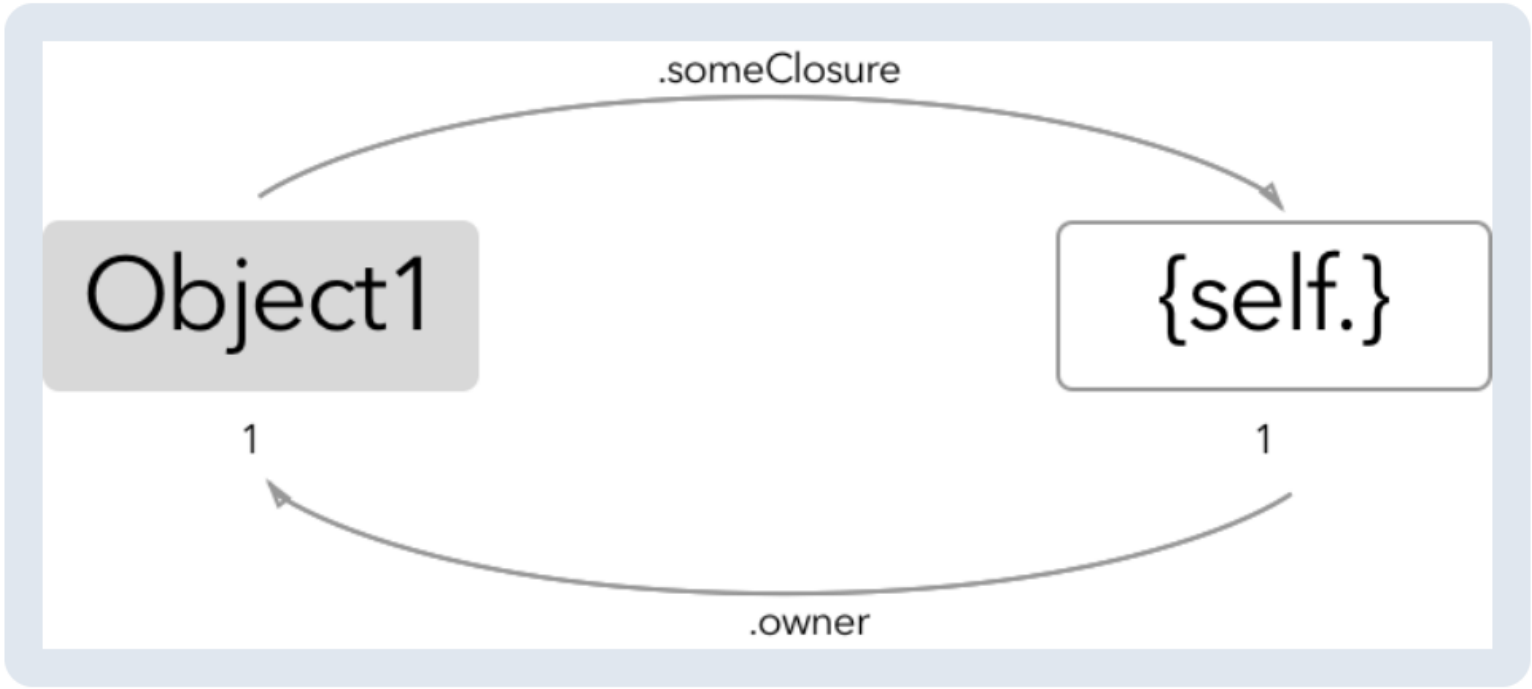
Memory leak

- Memory leak là việc bộ nhớ không thể được giải phóng khi object không còn được sử dụng.



Memory managment trong Closure

- Closure cũng như class là reference types



Closure capture list

- Capture list sẽ quy định luật để lấy giá trị của property trong closure

```
var x = 5
var y = 5

let someClosure = { [x] in
  print("\(x), \(y)")
}
x = 6
y = 6

someClosure()         // Prints 5, 6
print("\(x), \(y)")   // Prints 6, 6
```

Lập trình hướng đối tượng (OOP)

- OOP là một kỹ thuật lập trình cho phép dev tạo ra các đối tượng trong code, trừu tượng hóa các đối tượng trong thực tế.
- OOP có 4 tính chất cơ bản:
  - Tính trừu tượng
  - Tính kế thừa
  - Tính đa hình
  - Tính đóng gói

Tính trừu tượng

- Dữ liệu trừu tượng: là việc bỏ qua hay không chú ý đến một số khía cạnh của thông tin, chỉ tập trung vào những cốt lõi cần thiết.
- Lớp trừu tượng: một đối tượng ban đầu có thể có một số đặc điểm chung cho nhiều đối tượng khác như là sự mở rộng của nó nhưng bản thân đối tượng ban đầu này **có thể** không có các biện pháp thi hành

Tính kế thừa

- Cho phép một đối tượng có thể có sẵn các đặc tính mà đối tượng khác đã có thông qua kế thừa. Điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại

Tính đa hình

- Thể hiện thông qua việc gửi các **thông điệp** (*message*). Việc gửi các thông điệp này có thể so sánh như việc gọi các hàm bên trong của một đối tượng. Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới sẽ có phản ứng khác nhau.

Tính đóng gói

- Chỉ có các phương thức nội tại của đối tượng cho phép thay đổi trạng thái của nó. Việc cho phép môi trường bên ngoài tác động lên các dữ liệu nội tại của một đối tượng theo cách nào là hoàn toàn tùy thuộc vào người viết mã. Đây là tính chất đảm bảo sự toàn vẹn của đối tượng.