1. **Đặc điểm chung của lập trình mobile là gì? Yếu tố nào quyết định tới ứng dụng mobile?**

- thiết kế, xây dựng các ứng dụng chạy trên các hệ điều hành mobile android, ios

- phát triển cơ sở dữ liêuj cho app.

- yếu tố quyết định tới ứng dụng mobile: hệ điều hành, cơ sở dữ liệu, kiến trúc app.

1. **Liệt kê các cách passing data trong iOS & Swift?**

<https://techmaster.vn/posts/36328/truyen-du-lieu-trong-ios-swift>

- passing data: truyền dữ liệu

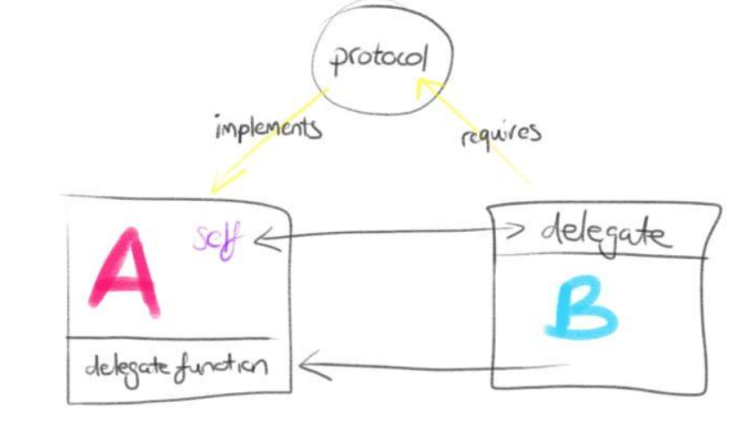
1. *passing data between viewcontrollers: bị ảnh hưởng bởi kiến trúc ứng dụng.*

+ Properties (A -> B): property là một variable of class. Bạn có thể gán giá trị cho variable và gán sang class B.



1. *Truyền data ngược lại giữa các ViewController với Properties (A <- B):*

- **Delegate**: Với delegation, một base class có thể chuyển chức năng cho một secondary class. Như vậy coder có thể implement secondary class này và phản hồi event cho base class bằng cách sử dụng protocol.



+ B1: định nghĩa một protocol bao gồm các chức năng mà chúng ta đang xử lý

+ delegate property là optional, và chúng ta sử dụng optional chaining khi gọi cookieWasBaked(\_:) function. Khi delegate là nil, thì function sẽ không được gọi.

B2: tao 1 class ke thua delegate protocol

+ UITableView class sử dụng UITableViewDelegate và UITableViewDataSource protocol.

+ CLLocationManager sử dụng CLLocationManagerDelegate để báo cáo dữ liệu liên quan đến vị trí cho ứng dụng của bạn, chặng hạn toạ độ GPS của iPhone.

+ UITextView sử dụng UITextViewDelegate để báo cho bạn về việc thay đổi trong text view.

+ Pros of delegate:

cách tiếp cận gọn nhẹ để chuyển giao các nhiệm vụ và tương tác từ class này sang class khác.

Chỉ cần một protocol để giao tiếp các yêu cầu giữa các class. Điều này làm giảm đáng kể việc ghép nối giữa các class.

Phân tách các nhiệm vụ của class tạo ra các tương tác với class phản hồi các nhiệm vụ này.

+ Một sai lầm điển hình của người mới bắt đầu là đặt ViewController làm delegate của mọi thứ. Một số lựa chọn thay thế:

- Tạo một class delegate riêng biệt, một controller, chịu trách nhiệm phản hồi các delegate function.

- Sử dụng các extension của Swift để phân tách code của bạn, tạo mỗi delegate là một extension

- Tóm tắt nhiều delegate function vào một controller (hoặc "manager") và sử dụng các closure function để phản hồi dữ liệu chi tiết mà bạn thực sự cần.

**- Closure:**

<https://fxstudio.dev/closure-trong-10-phut/#google_vignette>

Closure là một block code, có thể tách ra để tái sử dụng. Hiểu đơn giản hơn thì Closure là function, nhưng khuyết danh. Ta có thể gán Closure vào biến và sử dụng như các kiểu value khác.

+ Promise hoặc async / await là một cách tiếp cận hữu ích để quản lý nhiều async closure.

+ Sử dụng với cơ chế callback như delegate.

+ Quản lý memory khi sử dụng closure: using [weak self] or unowned.

+ Nguyen nhan cua leak memory of closure: Đó chính là khả năng Capturing Values của closure, mặc định nếu chúng ta không sử dụng weak hoăc unowned cho biến số bên ngoài khối closure thì closure sẽ tạo một strong reference đến biến số đó và chỉ giải phóng khi closure kết thúc.

**- NotificationCenter:**

+ It helps observing and broadcasting singles from a class to other classes.

+ 3 steps:

- observing notification: register notification with a key

- post notification: with the key

- responding to notification

+ Pros of NotificationCenter:

- passing data between classes which are not connected.

- ViewController can be not exist. It’s happened when calling Rest Api and receiving data before TableView is showed.

- Notification is many to many. Nhiều ViewController có thể phản hồi một notification, hoặc một ViewController muốn phản hồi nhiều notification.

+ Nếu bạn muốn sử dụng local traffic giữa các ViewController, thì không cần phải sử dụng NotificationCenter, đơn giản là bạn chỉ cần dùng delegate, property hoặc closure. Nhưng nếu bạn muốn liên tục và thường xuyên gửi dữ liệu từ một phần của ứng dụng này sang phần khác, NotificationCenter là một giải pháp tốt.

1. **Vì sao Swift được gọi là ngôn ngữ an toàn:**

- Swift :

+ doesn’t have undefined variables,

+ doesn’t have error of array out of bound,

+ doesn’t have integer overflow

+ handle nil, null value of variable

+ auto in manager memory.

1. **Generic là gì? Trình bày các loại cơ bản Generic trong Swift?**

<https://viblo.asia/p/generic-trong-swift-Ljy5VodkKra>

<https://fxstudio.dev/generics-trong-10-phut-swift/#google_vignette>

- Generic is data type which can represent/ replace for other data types

- **func** swapTwoValues**<**T**>(**\_ a: **inout** T, \_ b: **inout** T**)** **{}**

T is placeholder in generic.

- Generic type: type constraint,

+ type constraint:



1. **Trình bày khái niệm về Higher Order Function?**

- Higher order function has:

+ parameter can be value, object, another function.

+ return a function.

- Higher order function in swift:

+ map: convert a object to another

+ compactMap: is same map, one moreover, it handle optional element. CompactMap is used to filter nil values.

+ flatMap: = map + compactMap:

- convert value of elements

- handle element

- convert type of object

+ filter:

+ reduce: combining all elements in array to become a value.

+ sorted:

1. **Functional Programming:**

- Functional Programming: phương thức giải quyết vấn đề bằng cách chia nhỏ thành các function. Mỗi function lại có input là 1 function khác và output lại là 1 function.

1. **Trình bày về các cách mà bạn đã debug trong iOS & Swift?**

- print statements

- set breakpoints

- conditional breakpoints: when met the condition, breakpoints will be triggered.

- exception breakpoints: this can help to catch and diagnose crashes.

- visual debugging: view hierarchy of UI layout

- logging library:

- profiling tools: example instrument. Using this to analyze app performance, memory usage and energy consumption.