**Life cycle App AND Life ViewContrller trong swift**

1. **Life cycle App**

Vòng đời của một ứng dụng iOS là chuỗi các trạng thái mà ứng dụng trải qua từ lúc khởi động đến khi tắt. Hiểu rõ vòng đời của ứng dụng rất quan trọng để quản lý tài nguyên, thực hiện các hành động phù hợp vào các thời điểm khác nhau và đảm bảo ứng dụng hoạt động mượt mà. Dưới đây là các trạng thái chính của vòng đời ứng dụng iOS và các phương thức tương ứng trong AppDelegate để quản lý chúng.

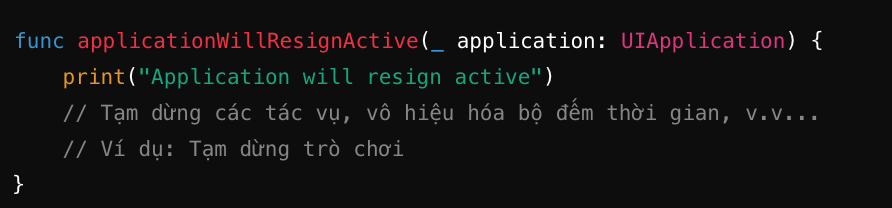
**Trạng thái của Ứng dụng**

**1. Not Running**

* Trạng thái này là khi ứng dụng chưa được khởi động hoặc đã bị hệ điều hành tắt hoàn toàn. Không có mã nào chạy trong trạng thái này, vì vậy không có ví dụ mã trực tiếp.

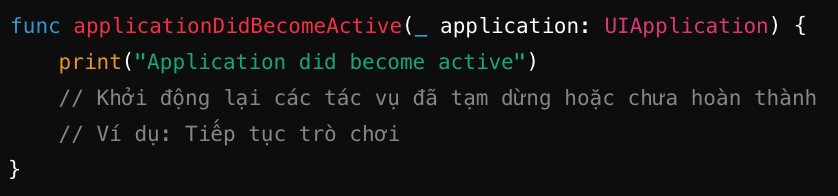
**2. Inactive**

* Ứng dụng có thể chuyển sang trạng thái Inactive khi có một cuộc gọi đến hoặc khi người dùng mở Notification Center hay Control Center.
* Được thể hiện qua phương thức **ApplicationWillResignActive** trong **AppDelegate**.



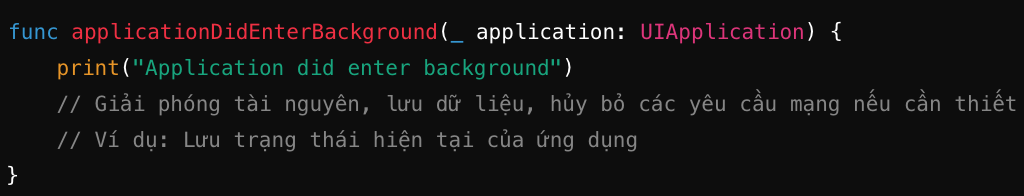
**3. Active**

* Khi ứng dụng trở lại trạng thái Active từ trạng thái Inactive hoặc Background, bạn có thể khởi động lại các tác vụ bị tạm dừng hoặc chưa hoàn thành.
* Được thể hiện qua phương thức **ApplicationDidBecomeActive** trong **AppDelegate**.



**4. Background**

* Ứng dụng chuyển sang trạng thái **Background** khi người dùng nhấn nút Home hoặc mở ứng dụng khác.
* Được thể hiện qua phương thức **ApplicationDidEnterBackground** trong **AppDelegate**.

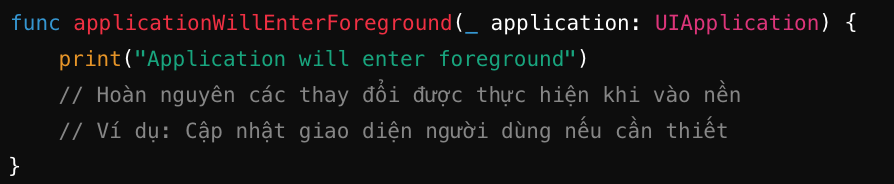


**5. Suspended**

* Ứng dụng chuyển sang trạng thái Suspended từ trạng thái Background sau một thời gian ngắn nếu không có công việc cần thực hiện. Trạng thái này không thể can thiệp trực tiếp, nhưng cần chuẩn bị cho nó trong **applicationDidEnterBackground**(\_:).

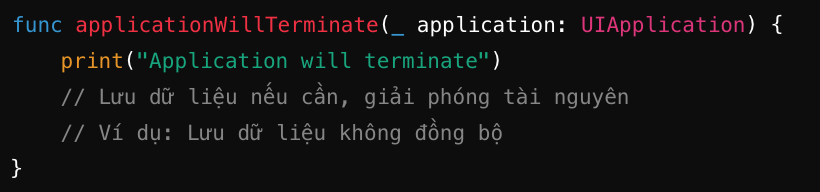
**6.** **Foreground (chuyển từ Background sang Active)**

* Khi ứng dụng chuyển từ Background sang Active, bạn cần chuẩn bị lại các tài nguyên và trạng thái.
* Được thể hiện qua phương thức **ApplicationWillEnterForeground** trong **AppDelegate**.



**7. Termination**

* Ứng dụng có thể bị tắt bởi hệ điều hành khi người dùng tắt ứng dụng hoặc hệ thống cần giải phóng bộ nhớ. Bạn nên lưu dữ liệu và giải phóng tài nguyên trong phương thức này.
* Được thể hiện qua phương thức **ApplicationWillTerminate** trong **AppDelegate**.



**Lưu Ý**

* **Lưu dữ liệu kịp thời**: Khi ứng dụng vào nền hoặc sắp bị tắt, đảm bảo lưu dữ liệu kịp thời để tránh mất dữ liệu.
* **Giải phóng tài nguyên**: Khi ứng dụng vào nền, giải phóng tài nguyên không cần thiết để tiết kiệm bộ nhớ và tài nguyên hệ thống.
* **Tái khởi động các tác vụ cần thiết**: Khi ứng dụng trở lại trạng thái hoạt động, tái khởi động các tác vụ bị tạm dừng để đảm bảo trải nghiệm người dùng không bị gián đoạn.

1. **Life cycle ViewController**

- Vòng đời của UIViewController trong iOS là quá trình từ khi một view controller được tạo ra đến khi nó bị giải phóng khỏi bộ nhớ. Hiểu rõ vòng đời này giúp bạn quản lý các tài nguyên, cập nhật giao diện người dùng, và thực hiện các tác vụ phù hợp ở từng giai đoạn. Dưới đây là các giai đoạn chính trong vòng đời của một UIViewController và những phương thức tương ứng bạn có thể ghi đè để tùy chỉnh hành vi của nó.

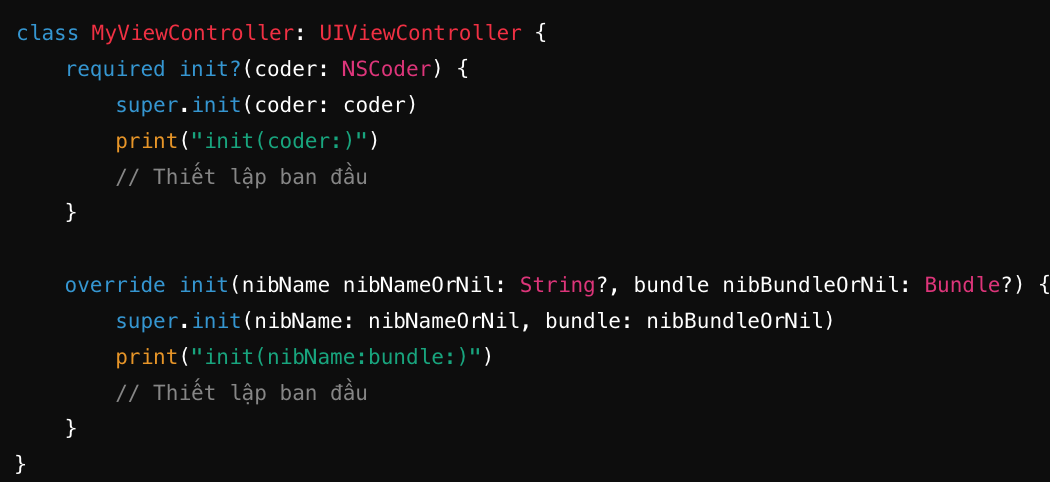
**Các Giai Đoạn Chính**

1. **Initialization** (Khởi tạo)
2. **View Loading** (Tải view)
3. **View Appearing** (Hiển thị view)
4. **View Running** (Chạy view)
5. **View Disappearing** (Ẩn view)
6. **View Destruction** (Hủy view)

**Các Phương Thức Tương Ứng:**

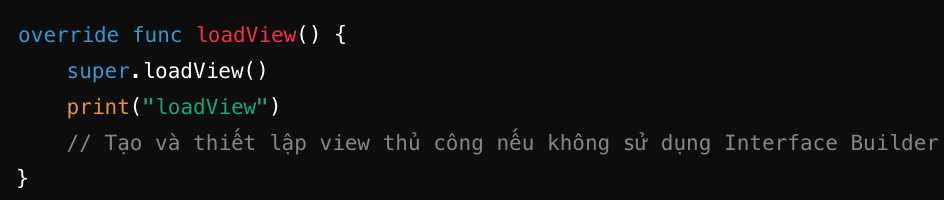
#### **Initialization (Khởi tạo)**

* **init(coder:) hoặc init(nibName:bundle:)**: Được gọi khi view controller được khởi tạo. Sử dụng phương thức này để thực hiện các thiết lập ban đầu. VD:

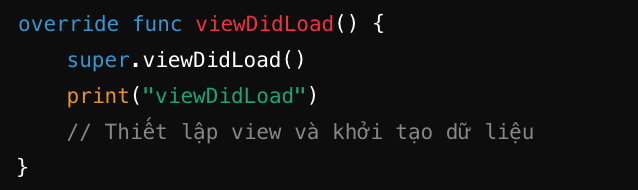
****

#### **View Loading (Tải view)**

* **loadView()**: Được gọi khi view của view controller chưa được tải vào bộ nhớ. Bạn có thể ghi đè phương thức này để tạo và thiết lập view thủ công. VD:

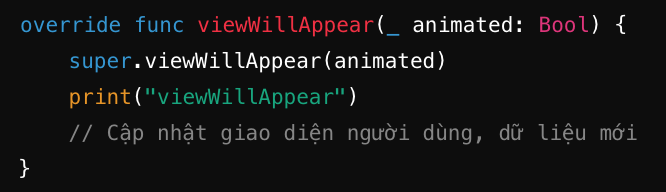
****

* **viewDidLoad()**: Được gọi khi view đã được tải vào bộ nhớ. Thực hiện các thiết lập bổ sung và khởi tạo dữ liệu trong phương thức này. VD:

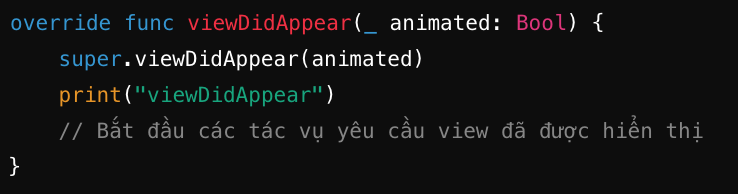


#### **View Appearing (Hiển thị view)**

* **viewWillAppear(\_:)**: Được gọi ngay trước khi view của view controller được thêm vào cây view và hiển thị cho người dùng. Phương thức này có thể được gọi nhiều lần trong vòng đời của view controller. VD:



* **viewDidAppear(\_:)**: Được gọi ngay sau khi view của view controller được thêm vào cây view và hiển thị cho người dùng. VD:

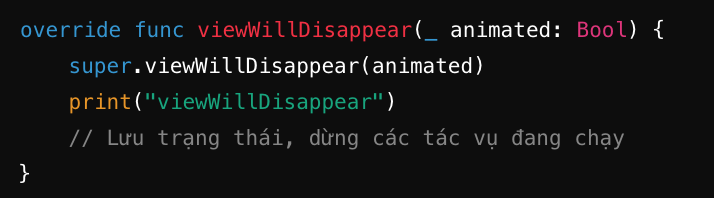
****

#### **View Running (Chạy view)**

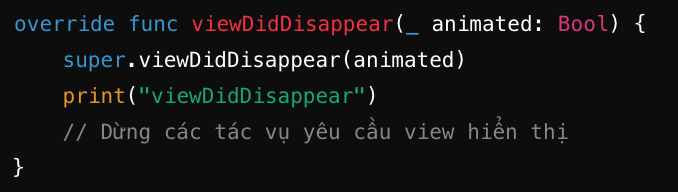
* Đây là giai đoạn view controller đang hoạt động bình thường, nhận sự kiện từ người dùng và tương tác với các thành phần giao diện.

#### **View Disappearing (Ẩn view)**

* **viewWillDisappear(\_:)**: Được gọi ngay trước khi view của view controller bị xóa khỏi cây view và không còn hiển thị cho người dùng. Phương thức này có thể được gọi nhiều lần trong vòng đời của view controller. VD:



* **viewDidDisappear(\_:)**: Được gọi ngay sau khi view của view controller bị xóa khỏi cây view và không còn hiển thị cho người dùng. VD:

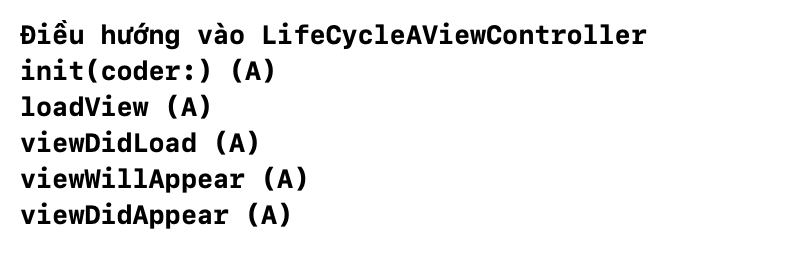


#### **View Destruction (Hủy view)**

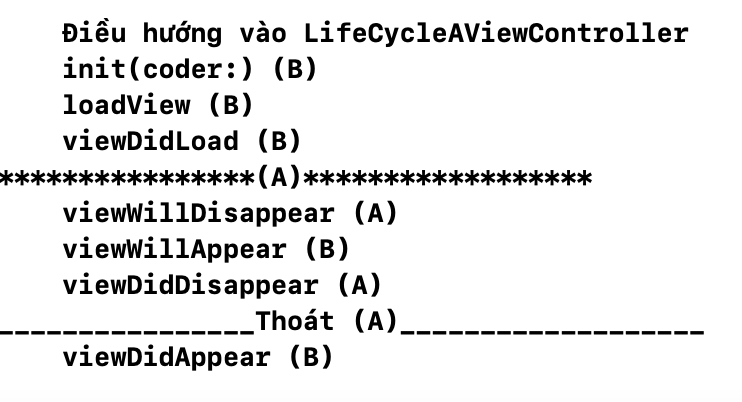
* **deinit**: Được gọi ngay trước khi view controller bị giải phóng khỏi bộ nhớ. Sử dụng phương thức này để giải phóng tài nguyên. VD:



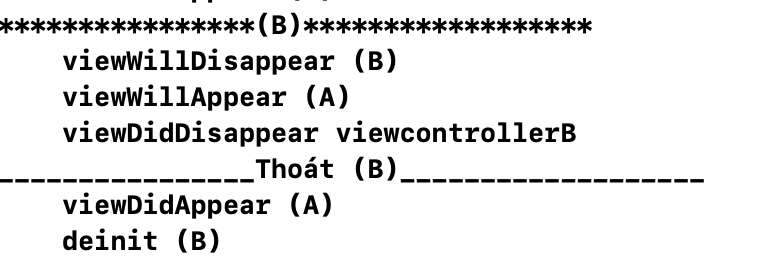
**VD:** Tôi xây dựng 2 **UIViewController** là **LifeCycleAViewController** và **LifeCycleBViewController**, tôi lần lượt triển khai các phương thức với một yêu cầu in ra màn hình thông báo khi thực hiện phương thức nào để biết khi nào phương thức nào được thực hiển và thứ tự thực hiện của các phương thức.  
  
Thực tế tôi sẽ thực hiện như sau.  
1: Khi tôi điều hướng từ viewHome vào **LifeCycleAViewController** sẽ được in ra kết quả sau:



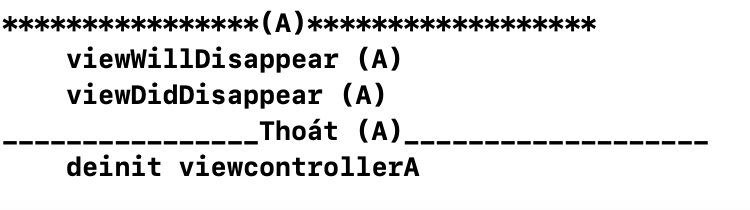
2: Tiếp theo khi tôi điều hướng từ **LifeCycleAViewController** đến **LifeCycleBViewController**



3: Tôi back lại **LifeCycleAViewController**



4: Tôi back lại viewHome ban đầu



Kết luận: Từ ví dụ trên ta có thể thấy được thứ tự xảy ra các sự kiện kể từ lúc bắt đầu khởi tạo điều hướng vào ViewController đến khi thoát khởi ViewController đó, tùy theo yêu cầu mà chũng ta có thể tùy chỉnh những công việc phù hợp với từng tác vụ phù hợp với từng vòng đời của viewController để mang lại hiệu quả tốt nhất.