Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình iOS

Bài 3. *Quản lý bộ nhớ trong Objective-C*

Ngành Mạng & Thiết bị di động

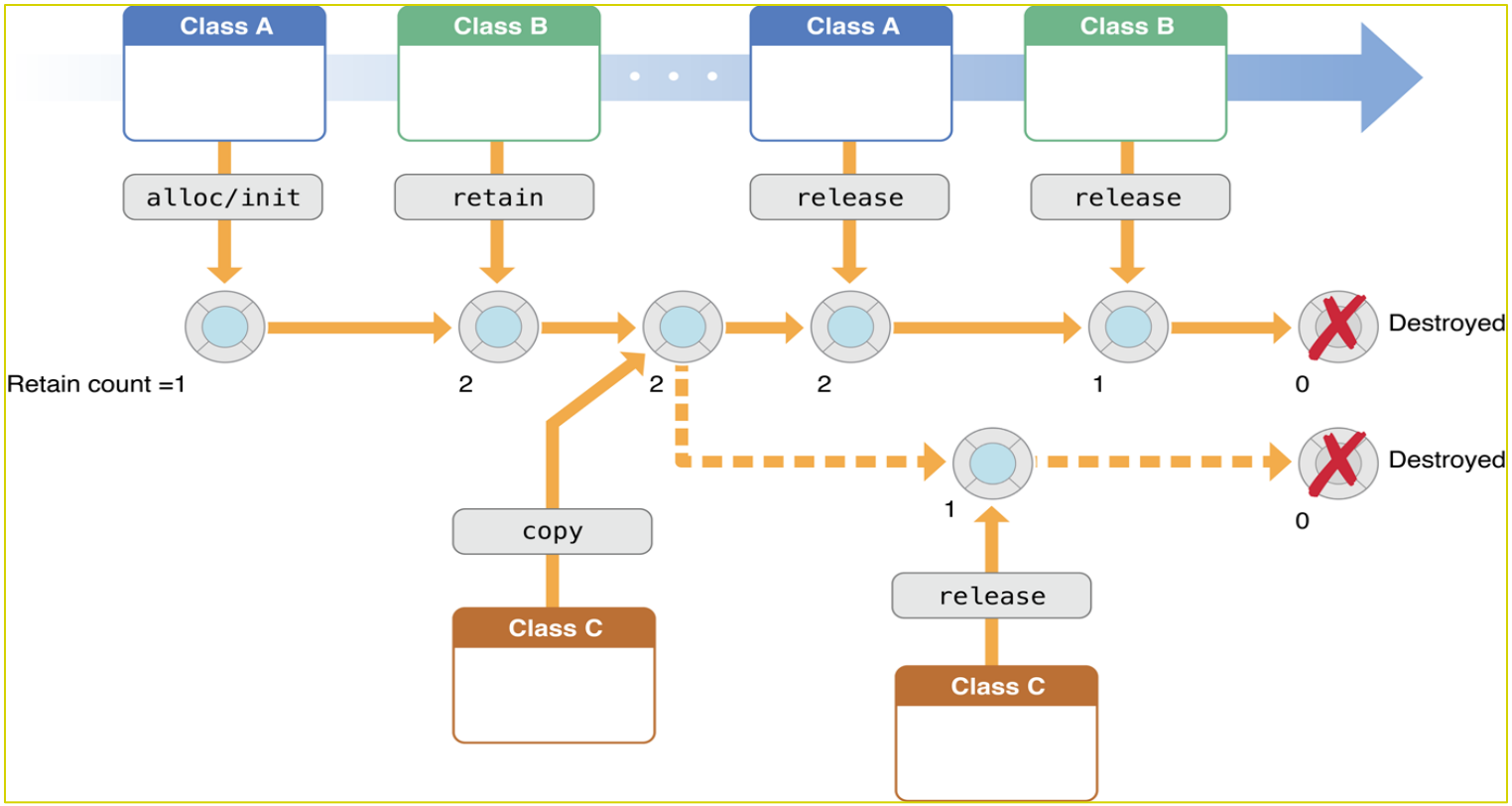
2015

Nội dung

1. Tổng quan quản lý bộ nhớ
2. Các quy tắc trong quản lý bộ nhớ
3. Cách quản lý bộ nhớ hiệu quả
4. Sử dụng AutoRelease Pool Block

1 Tổng quan quản lý bộ nhớ

- Quản lý bộ nhớ là quá trình cấp phát bộ nhớ khi sử dụng chương trình và giải phóng bộ nhớ khi không dùng nữa.



1 Tổng quan quản lý bộ nhớ

- **Trong Objective-C, có hai cách để quản lý bộ nhớ của ứng dụng:**
  - Manual Retain-Release (MRR), ta quản lý bằng cách theo dõi các đối tượng đã được tạo ra.
  - Automatic Reference Counting (ARC), hệ thống sử dụng cùng "reference counting" như MRR, nhưng nó có kèm theo các phương thức quản lý bộ nhớ vào lúc biên dịch.

- **Có hai vấn đề chính trong việc quản lý bộ nhớ không đúng:**
  - Giải phóng hoặc ghi đè dữ liệu trong khi đang sử dụng. Việc này làm cho ứng dụng bị lỗi, hoặc tệ hơn là mất dữ liệu.
  - Không giải phóng dữ liệu khi không cần thiết, gây ra việc rò rỉ bộ nhớ. Việc rò rỉ làm cho bộ nhớ của ứng dụng ngày càng tăng, làm cho hiệu suất hệ thống giảm hoặc ứng dụng bị lỗi.

Nội dung

1. Tổng quan quản lý bộ nhớ
2. **Các quy tắc trong quản lý bộ nhớ**
3. Cách quản lý bộ nhớ hiệu quả
4. Sử dụng AutoRelease Pool Block

2 Các qui tắc trong quản lý bộ nhớ

- **Mô hình quản lý bộ nhớ dựa trên quyền sở hữu đối tượng. Một đối tượng bất kỳ có thể có một hoặc nhiều chủ sở hữu. Nếu đối tượng không có chủ sở hữu, khi chạy hệ thống sẽ tự động hủy đối tượng đó. Cocoa có đặt vài quy tắc sau:**
  - Ta sở hữu bất kỳ đối tượng nào do ta tạo ra. .
  - Lấy quyền sở hữu của đối tượng bằng cách dùng **retain** .
  - Khi không cần sử dụng nữa, ta phải từ bỏ quyền sở hữu đối tượng.
  - Ta không được phép từ bỏ quyền sở hữu đối tượng khi không làm chủ.
  - Ta không sở hữu đối tượng được trả về bằng tham chiếu.

2 Các qui tắc trong quản lý bộ nhớ

- **Ví dụ:** Ta có lớp Person như sau:

```
@interface Person : NSObject
@property (retain) NSString *firstName;
@property (retain) NSString *lastName;
@property (assign, readonly) NSString *fullName;
@end
```

Ta sẽ sử dụng lớp Person như sau:

```
Person *aPerson = [[Person alloc] init];
// ...
NSString *name = aPerson.fullName;
// ...
[aPerson release];
```

2 Các qui tắc trong quản lý bộ nhớ

- **Ví dụ sử dụng autorelease:**

```
- (NSString *)fullName {
  NSString *string = [[[NSString alloc]
initWithFormat:@"%@ %@", self.firstName, self.lastName]
autorelease];
  return string;
}
```

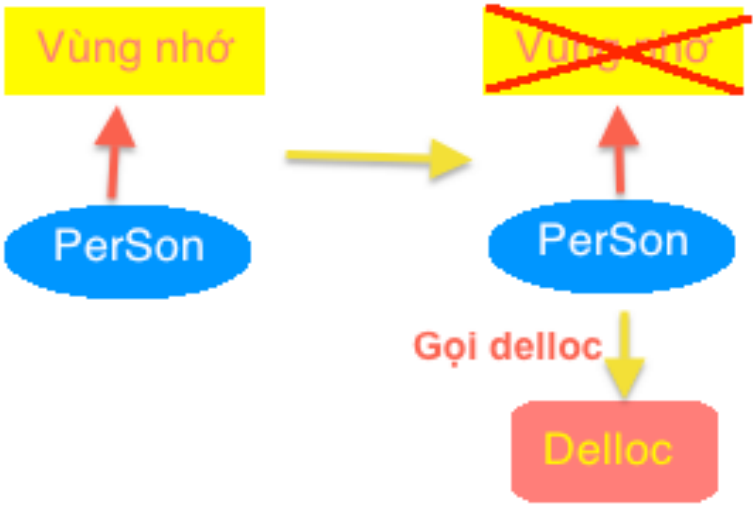
- **Ví dụ không sử dụng autorelease:**

```
- (NSString *)fullName {
  NSString *string = [NSString stringWithFormat:@"%@
%@", self.firstName, self.lastName];
  return string;
}
```

2 Các qui tắc trong quản lý bộ nhớ

- Lớp NSObject định nghĩa phương thức dealloc, được tự động gọi khi một đối tượng không có chủ sở hữu và bộ nhớ của nó thì bị thu hồi. Nhiệm vụ của phương thức dealloc là làm trống bộ nhớ của đối tượng và hủy bỏ tất cả tài nguyên mà nó đang giữ :

```
- (void)dealloc
{
    [_firstName release];
    [_lastName release];
    [super dealloc];
}
```



Nội dung

1. Tổng quan quản lý bộ nhớ
2. Các quy tắc trong quản lý bộ nhớ
3. **Cách quản lý bộ nhớ hiệu quả**
4. Sử dụng AutoRelease Pool Block

3 Cách quản lý bộ nhớ hiệu quả

- Sử dụng các phương thức truy xuất làm cho việc quản lý bộ nhớ dễ dàng hơn.
- Sử dụng các phương thức truy xuất để cài đặt giá trị cho thuộc tính.
- Không sử dụng các phương thức truy xuất trong phương thức khởi tạo và dealloc.
- Sử dụng tham chiếu yếu để tránh vòng chiếm giữ.

3 Cách quản lý bộ nhớ hiệu quả

- Sử dụng các phương thức truy xuất làm cho việc quản lý bộ nhớ dễ dàng hơn.

```
@interface Counter : NSObject
@property (nonatomic, retain) NSNumber *count;
@end
```

- Phương thức getter

```
- (NSNumber *)count{
  return _count;
}
```

- Phương thức setter

```
- (void)setCount:(NSNumber
*)newCount{
  [newCount retain];
  [_count release];
  _count = newCount;
}
```

3 Cách quản lý bộ nhớ hiệu quả

- **Sử dụng các phương thức truy xuất để cài đặt giá trị cho thuộc tính.**
  - Tạo một NSNumber với phương thức alloc, và sẽ release nó.

```
- (void)reset {
  NSNumber *zero = [[NSNumber alloc] initWithInteger:0];
  [self setCount:zero];
  [zero release];
}
```

  - Sử dụng hàm khởi tạo để tạo đối tượng NSNumber mới. Ta sẽ không cần retain và release đối tượng.

```
- (void)reset {
  NSNumber *zero = [NSNumber numberWithInteger: 0];
  [self setCount:zero];
}
```

3 Cách quản lý bộ nhớ hiệu quả

- **Không sử dụng các phương thức truy xuất trong phương thức khởi tạo và dealloc.**
  - Trong phương thức khởi sẽ có một số thuộc tính được tạo sau hàm khởi tạo nếu ta truy xuất có thể dẫn đến lỗi và không tạo được đối tượng.
  - Dealloc là phương thức dùng để dọn dẹp vùng nhớ của đối tượng nếu ta dùng các phương thức truy xuất trong hàm này nó có thể phát sinh vùng nhớ khi đối tượng bị huỷ và đây là mối nguy hiểm không nên có.

3 Cách quản lý bộ nhớ hiệu quả

- Ví dụ phương thức khởi tạo(init):

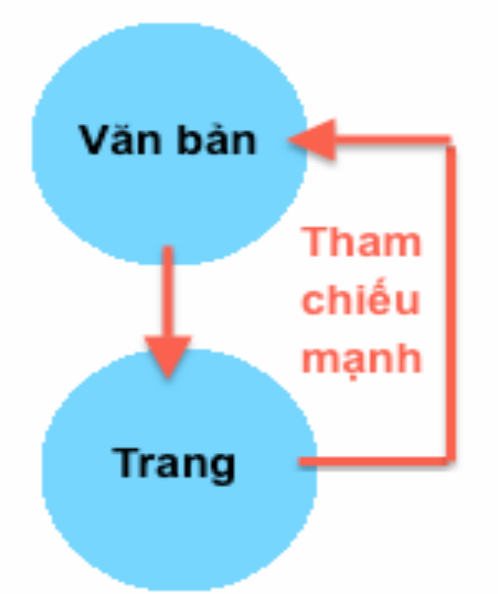
```
- init {
  self = [super init];
  if (self) {
    _count = [[NSNumber alloc] initWithInteger:0];
  }
  return self;
}
```

- Ví dụ phương thức hủy(dealloc):

```
- (void)dealloc {
    [_count release];
    [super dealloc];
}
```

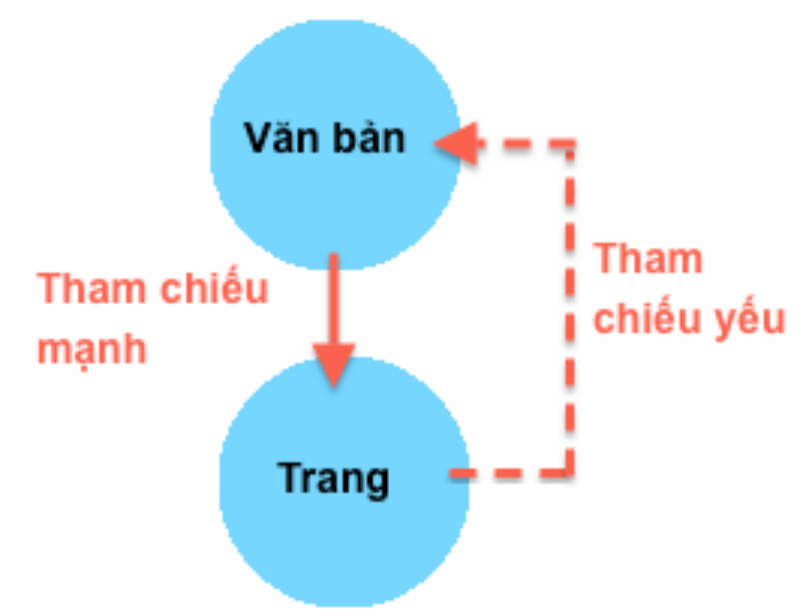
3 Cách quản lý bộ nhớ hiệu quả

- **Sử dụng tham chiếu yếu để tránh vòng chiếm giữ.**
  - Chiếm giữ một đối tượng tạo nên một tham chiếu mạnh. Một đối tượng không thể hủy cho tới khi tất cả tham chiếu mạnh đều được giải phóng.
  - **Retain Cycle** (vòng chiếm giữ) xảy ra nếu có hai đối tượng tham chiếu mạnh tới nhau



3 Cách quản lý bộ nhớ hiệu quả

- **Sử dụng tham chiếu yếu để tránh vòng chiếm giữ.**
  - Để giải quyết vấn đề cho vòng chiếm giữ, ta dùng tham chiếu yếu. Tham chiếu yếu sẽ không sở hữu đối tượng.
  - Để giữ sơ đồ đối tượng, ta cần phải có tham chiếu mạnh (nếu chỉ có tham chiếu yếu, các đối tượng sẽ không có chủ sở hữu sẽ bị hủy).



Nội dung

1. Tổng quan quản lý bộ nhớ
2. Các quy tắc trong quản lý bộ nhớ
3. Cách quản lý bộ nhớ hiệu quả
4. **Sử dụng AutoRelease Pool Block**

4 Sử dụng AutoRelease Pool Block

- Autorelease pool block cung cấp cơ chế mà ở đó ta có thể từ bỏ quyền sở hữu đối tượng và tránh được việc nó bị hủy ngay lập tức. Thông thường ta không cần phải tạo Autorelease pool block, nhưng trong một số tình huống ta phải tạo hoặc do nó có lợi.
- Một autorelease pool block được đánh dấu bởi **@autoreleasepool**, ví dụ:

```
@autoreleasepool {
        // Lệnh để tạo các đối tượng autorelease.
}
```

4 Sử dụng AutoRelease Pool Block

- Cuối autorelease pool block, các đối tượng đã nhận một thông điệp autorelease đều được gửi thông điệp release.

- Autorelease pool block có thể lồng vào nhau.

```
@autoreleasepool {
        // . . .
        @autoreleasepool {
                // . . .
        }
        // . . .
}
```

4 Sử dụng AutoRelease Pool Block

- Cocoa luôn muốn lệnh được thực thi bên trong một autorelease pool block, nếu không các đối tượng được autorelease sẽ không được giải phóng và làm cho ứng dụng bị rò rỉ bộ nhớ.

- Framework AppKit và UIKit xử lý mỗi sự kiện bên trong một autorelease pool block, do đó thông thường ta không cần phải tạo autorelease pool block hay thấy được lệnh sử dụng nó.

4 Sử dụng AutoRelease Pool Block

- **Có ba trường hợp ta có thể sử dụng autorelease pool block:**
  - Khi đang viết một chương trình không dựa trên framework UI như công cụ command-line.
  - Viết một vòng lặp để tạo nhiều đối tượng tạm thời.
  - Nếu ta tạo ra một tiến trình thứ hai.

Thảo luận