



Lập trình iOS

Bài 3. Lập trình hướng đối tượng trong Objective - C

Ngành Mạng và Thiết bị di động





Nội dung

1. Khái niệm về lập trình đối tượng trong Objective-C

1. Lớp (Class)
2. Thuộc tính (Property)
3. Phương thức (Method)
4. Đối tượng (Object)

2. Các tính chất của hướng đối tượng trong Objective-C

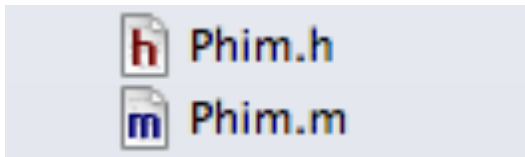


1.1 Lớp (Class)

- ❑ Lớp là một khái niệm trừu tượng về đối tượng. Nó quy định các thuộc tính và phương thức mà đối tượng có thể có.
- ❑ Trong objective-C lớp được định nghĩa thành 2 phần tương tự như C/C++ là tập tin *.h và *.m:
 - Tập tin *.h: dùng để khai báo các thuộc tính (các biến toàn cục), phương thức (hàm) có trong lớp.
 - Tập tin *.m: sử dụng định nghĩa hoặc triển khai các thuộc tính, phương thức đã được khai báo trong tập tin *.h
- ❑ Những thuộc tính và phương thức không được khai báo trong tập tin *.h thì khi sử dụng lớp, sẽ không thể truy xuất đến các thuộc tính và phương thức đó



1.1 Lớp (Class)



```
#import <Foundation/Foundation.h>
//Import các tập tin/thư viện cần thiết
@interface Nguoi : NSObject
{
    //Khai báo các thuộc tính
}
//Khai báo các thuộc tính sử dụng @property
    và phương thức của lớp
@end
```

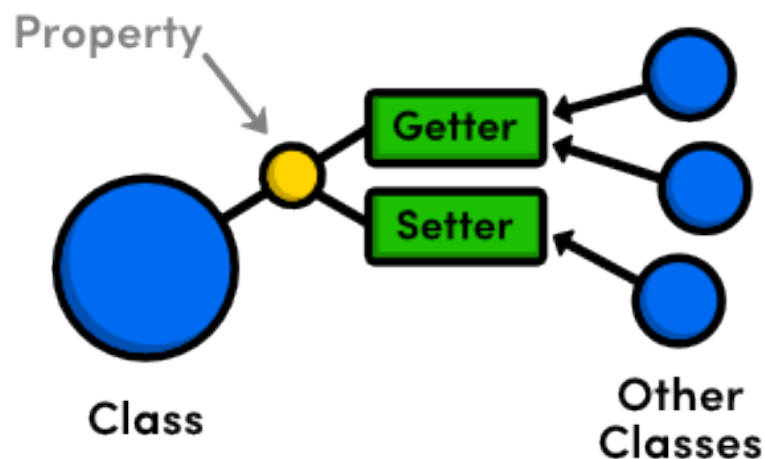
```
#import "Nguoi.h"
//Import các tập tin/thư viện cần thiết
@interface Nguoi()
//Sử dụng tương tự @interface bên tập tin Nguoi.h
//Chỉ có thể sử dụng trong tập tin này
@end

@implementation Nguoi{
    //Khai báo các thuộc tính
    //Chỉ có thể sử dụng trong tập tin này
}
//Thực thi các phương thức đã khai báo
@end
```



1.2. Thuộc tính (Property)

- ❑ Thuộc tính là thực chất là các biến toàn cục, có thể được sử dụng thông qua các phương thức đọc và viết. Các phương thức này sẽ đảm bảo tính bảo mật và đóng gói cho đối tượng.
- ❑ Thuộc tính sẽ giúp mô tả đối tượng có những tính chất gì. Ví dụ: Người có thuộc tính tên, tuổi, giới tính...





1.2. Thuộc tính (Property)

- ❑ Trong một lớp ta có thể quy định quyền hạn sử dụng thuộc tính dựa vào các loại từ khóa sau:
 - **Private:** Thuộc tính chỉ có thể sử dụng trong Lớp
 - **Protected:** Thuộc tính chỉ có thể sử dụng trong Lớp hoặc Lớp Con kế thừa từ Lớp. Từ khóa mặc định cho các thuộc tính không khai báo từ khóa.
 - **Public:** Thuộc tính có thể được truy xuất thông qua đối tượng tại bất kỳ lớp nào.
 - **Package:** Thuộc tính có thể được truy xuất ở bất cứ lớp nào thông qua một đối tượng cụ thể, tuy nhiên phải thuộc cùng gói ứng dụng hoặc thư viện.



1.2. Thuộc tính (Property)


❑ Cú pháp khai báo thuộc tính:

@{Từ khóa} {Kiểu dữ liệu} {Tên thuộc tính}


- Ví dụ khai báo thuộc tính cho lớp `Nguoi` như sau:

```
#import <Foundation/Foundation.h>

@interface Nguoi : NSObject{
    @private NSString *gioiTinh;
    @protected NSNumber *tuoi;
    @public NSString *ten;
}
@end
```



```
9 #import "Nguoi.h"
10
11 @implementation Nguoi{
12     @public NSNumber *soLuong;
13 }
14 @end
15
16
```



- Để sử dụng các thuộc tính này ta sử dụng toán tử con trỏ “->” :

```
- (void)chaoHoi{
    Meo *thuCung = [[Meo alloc]init];
    NSLog(@"Xin chào, tôi là %@ và em mèo cưng %@",self->ten,
    thuCung->tenMeo);
}
```



1.2. Thuộc tính (Property)

- ❑ Cú pháp để khai báo thuộc tính bằng @property như sau:

`@property {Tập thuộc tính} {Kiểu dữ liệu} {Tên thuộc tính}`

- ❑ Trong đó:

- @property: là từ khóa bắt buộc
- Tập thuộc tính: là những khai báo về quản lý bộ nhớ cũng như quản lý truy xuất
- Kiểu dữ liệu: NSString, NSNumber, NSInteger, NSArray...
- Tên thuộc tính: ký tự đầu tiên phải viết chữ thường và thường là các danh từ: soHocSinh, diemThi, hoTen,...



1.2. Thuộc tính (Property)

❑ Tập thuộc tính bao gồm các dạng sau:

- Atomicity:
 - nonatomic: chỉ định thuộc tính có thể trả về các giá trị khác nhau ở thời điểm khác nhau trong các tiến trình khác nhau.
- Setter Semantic:
 - retain: giá trị thuộc tính nhận vào được lưu lại, giá trị lưu trữ trước đó được giải phóng khỏi vùng nhớ.
 - copy: sao chép giá trị thuộc tính mới nhận vào, giá trị lưu trữ trước đó được giải phóng khỏi vùng nhớ.
 - assign: thiết lập giá trị cho thuộc tính không sử dụng retain và copy.
 - strong: cùng ý nghĩa với retain nhưng sử dụng trong hệ thống ARC.
 - weak: cùng ý nghĩa với assign nhưng sử dụng trong hệ thống ARC.



1.2. Thuộc tính (Property)

- Read/write:
 - `readonly`: chỉ định thuộc tính có thể đọc và ghi dữ liệu, thuộc tính cần có phương thức đóng gói (`getter` & `setter`).
 - `read-only`: chỉ định thuộc tính chỉ có thể đọc, thuộc tính cần có phương thức đóng gói (`getter`).
- Methods Name:
 - `getter=<getterName>`: thay đổi tên phương thức `getter` thành `<getterName>`.
 - `setter=<setterName>`: thay đổi tên phương thức `setter` thành `<setterName>`.



1.2. Thuộc tính (Property)

- ❑ Để có thể sử dụng **property** cho các thuộc tính đã khai báo quyền truy xuất thì ta cần phải khai báo **synthesize** cho các thuộc tính trong **@implementation**

```
@interface Nguoi : NSObject{
@private NSString* gioiTinh;
@protected NSNumber* tuoi;
@public NSString *ten;
}
@property (nonatomic) NSString* gioiTinh;
@end

11
12 @implementation Nguoi
13 @synthesize gioiTinh;
14
15 @end
16
17
18
```



1.2. Thuộc tính (Property)

❑ Để sử dụng **property** ta có thể sử dụng các cách sau:

- Cách 1: Truy xuất trực tiếp

```
self->gioiTinh = @"Nam";  
NSLog(@"Giới tính của tôi là %@", self->gioiTinh);
```

- Cách 2 (Khuyến khích sử dụng): Truy xuất thông qua bộ getter - setter thuộc tính:

```
self.gioiTinh = @"Nữ"; //tự động gọi phương thức setter  
NSLog(@"Giới tính của tôi là %@", self.gioiTinh); //tự động  
gọi phương thức getter
```

- Cách 3 (Khuyến khích sử dụng): sử dụng phương thức getter – setter

```
[self setGioiTinh:@"Khó nói"]; //Phương thức setter  
NSLog(@"Giới tính của tôi là %@", [self gioiTinh]); //Phương  
thức getter
```



1.2. Thuộc tính (Property)

- ❑ Ngoài ra ta có thể sử dụng `@property` mà không khai báo quyền truy xuất thì nếu khai báo ở tập tin `.h` quyền truy xuất của thuộc tính sẽ là `Public`, còn ở tập tin `.m` thì quyền truy xuất là `Private`:

```
@interface Nguoi : NSObject{
    @private NSString *gioiTinh;
    @protected NSNumber *tuoi;
}
@property (nonatomic) NSString* ten;
@end
```

Public



```
12 @interface Nguoi()
13 @property (nonatomic) NSNumber* soLuong;
14 @end
15
16 @implementation Nguoi
17
18 @end
```

Private



→ khi truy xuất đến các thuộc tính này ta chỉ có thể sử dụng toán tử “.” hoặc phương thức getter và setter mà không thể truy xuất trực tiếp đến thuộc tính thông qua toán tử con trỏ “->”.



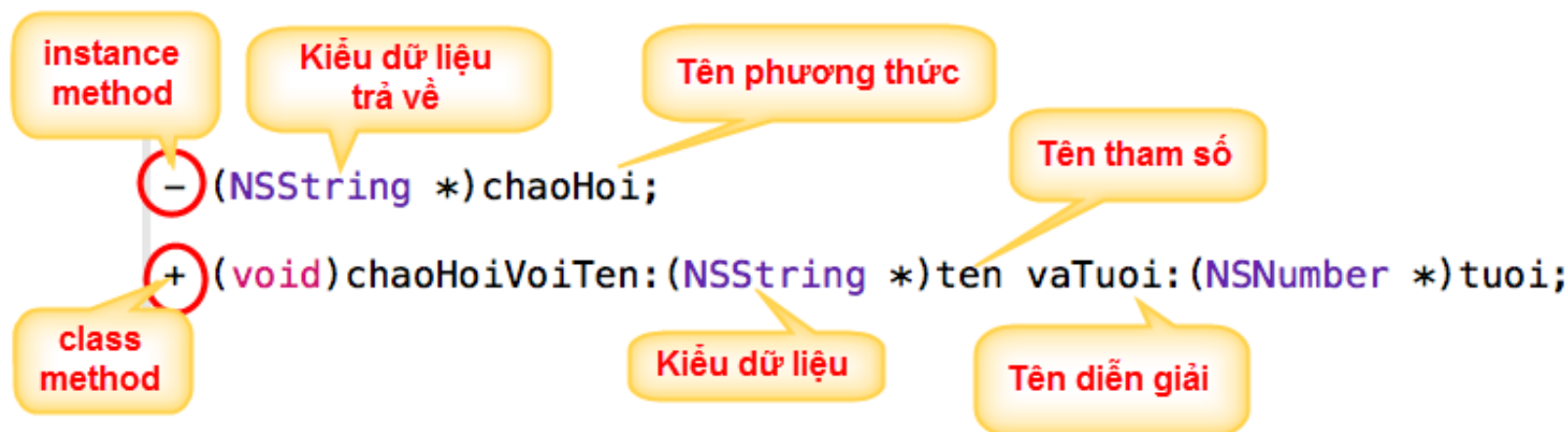
1.3. Phương thức (Method)

❑ Phương thức thực chất là hàm, dùng để xử lý các hành động của đối tượng.

❑ Để khai báo phương thức ta thực hiện theo cú pháp sau:

{Kiểu phương thức} ({Kiểu dữ liệu trả về}){Tên phương thức}: ({Kiểu dữ liệu 1}) {Tên tham số 1} ... {Tên biến giải tham số n}: ({Kiểu dữ liệu n}) {Tên tham số n};

❑ Ví dụ:





1.3. Phương thức (Method)

- ❑ Để triển khai phương thức ta thực hiện tại tập tin “*.m” như sau:

```
@implementation Ngươi
- (NSString *)chaoHoi{
    return [NSString stringWithFormat:@"Xin chào tôi tên là %@,
    %@ tuổi", self.hoTen, self.tuoi];
}
+ (void)chaoHoiVoiTen:(NSString *)ten vaTuoi:(NSNumber *)tuoi{
    NSLog(@"Xin chào tôi tên là %@, %@ tuổi",ten,tuoi);
}
@end
```



1.3. Phương thức (Method)

❑ Để gọi phương thức ta sử dụng cú pháp sau:

- Phương thức instance: [{tên đối tượng} {tên phương thức}:{giá trị truyền vào}];
- Phương thức class: [{tên lớp} {tên phương thức}:{giá trị truyền vào}];

```
//Gọi phương thức instance
NSString *loiChao = [self chaoHoi];
//Gọi phương thức class
[Nguoi chaoHoiVoiTen: @"Nguyễn Văn A" vaTuoi:@20];
```




1.4. Đối tượng (object)

- ❑ Đối tượng là một chủ thể xác định của lớp
- ❑ Đối tượng cho biết cụ thể những thông tin đó là gì và ta có thể thông qua đối tượng để sử dụng các hành động mà lớp đã xây dựng

- ❑ Cú pháp khởi tạo:

`<Tên Lớp> *<Tên đối tượng> = [[<Tên Lớp> alloc] init];`

- ❑ Ví dụ:

`Phim *phim = [[Phim alloc] init];`

- Ý nghĩa:

- Đối tượng phim được cấp phát bộ nhớ lưu trữ bằng phương thức **alloc**, sau đó khởi tạo giá trị mặc định thông qua phương thức **init**.



1.4. Đối tượng (object)


- ❑ Để sử dụng được lớp `Nguoi` tại lớp `ViewController` ta cần import lớp `Nguoi` như sau:

```
#import "ViewController.h"
#import "Nguoi.h"
@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    Nguoi *t = [[Nguoi alloc] init];
    t.hoTen = @"Trung";
    t.tuoi = @16;
    NSString *loiChao = [t chaoHoi];
    NSLog(@"%@", loiChao);
}
```





Nội dung

1. Khái niệm về lập trình đối tượng trong Objective-C
2. Các tính chất của hướng đối tượng trong Objective-C
 1. Tính kế thừa
 2. Tính đóng gói
 3. Tính đa hình



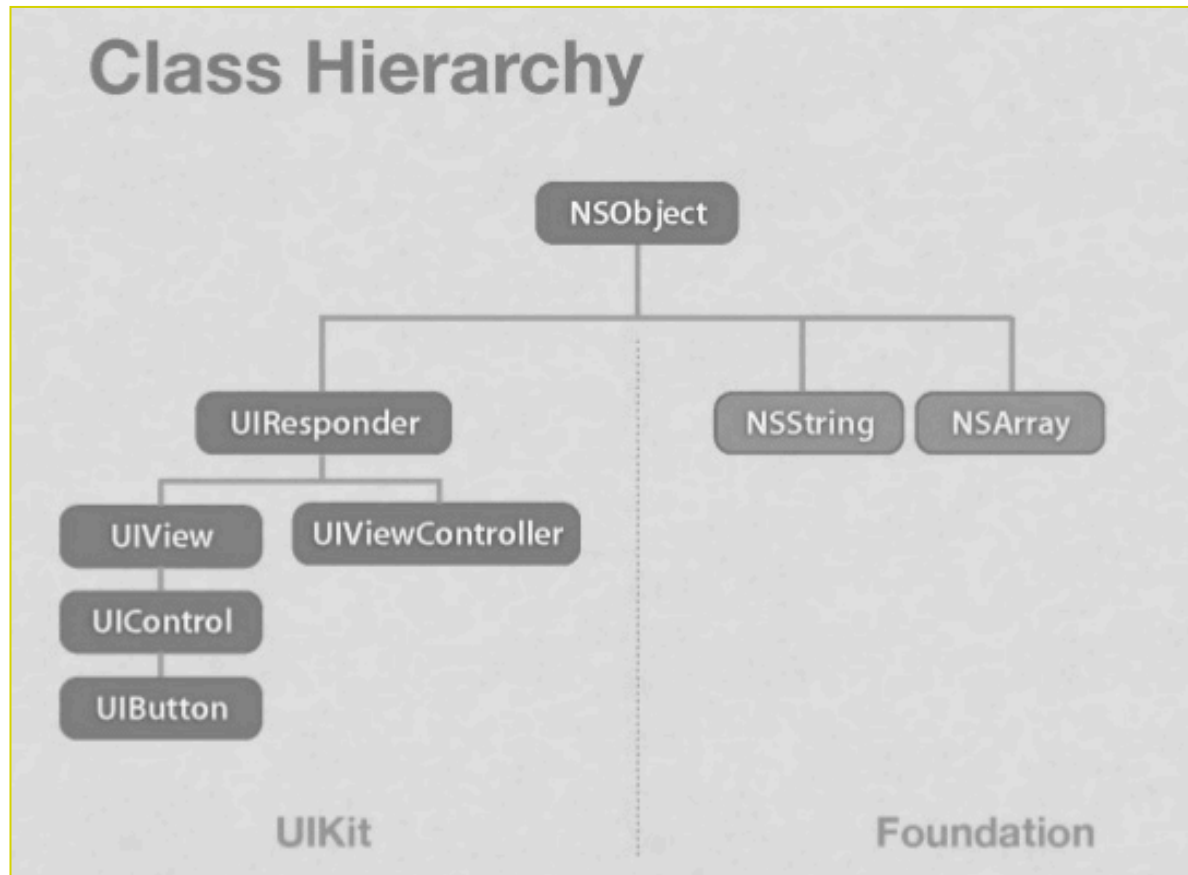
2.1. Tính kế thừa

- ❑ Tính kế thừa là một tính chất khá quan trọng trong lập trình hướng đối tượng, nó giúp tối giảm việc trùng lặp mã lệnh xảy ra ở các lớp, việc quản lý mã lệnh trở nên dễ dàng và thuận tiện hơn.
- ❑ Mỗi lớp chỉ kế thừa 1 lớp duy nhất. Một lớp thì có thể được kế thừa bởi nhiều lớp. Nghĩa là một lớp cha thì có thể có nhiều lớp con. Nhưng một lớp con thì chỉ có duy nhất một lớp cha.



2.1. Tính kế thừa

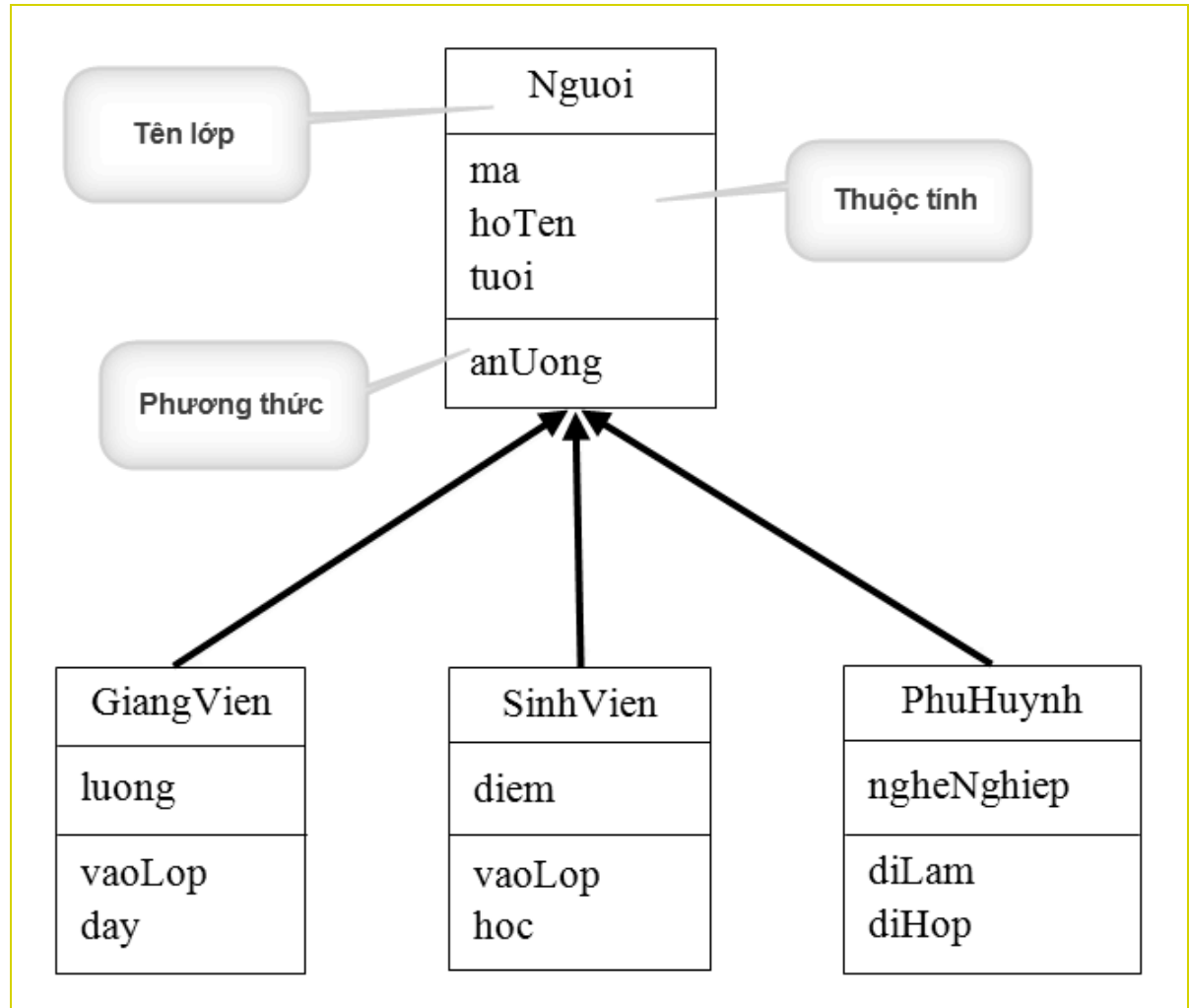
- ❑ Trong Objective C, tất cả các lớp đều được kế thừa từ lớp NSObject.





2.1. Tính kế thừa

- ❑ Ví dụ: ta có 3 lớp mô tả 3 loại đối tượng Giảng viên, Sinh viên và Phụ huynh





2.1. Tính đóng gói

- ❑ Tính đóng gói trong hướng đối tượng giúp cho thông tin được bảo mật.
- ❑ Việc truy xuất dữ liệu đều phải thông qua các phương thức getter và setter.
- ❑ Trong Objective C, khi khai báo thuộc tính bằng khóa `@property` thì hệ thống đã tạo sẵn các phương thức getter và setter mặc định, ta có thể chỉnh sửa nội dung của các phương thức đó theo ý mình.

```
@property (nonatomic, strong, readonly, getter=chaoHoi)  
NSString *hoTen;
```



2.1. Tính đa hình

- ❑ Khả năng tự định nghĩa cùng một phương thức ở các lớp đối tượng khác nhau theo một cách riêng của lớp đối tượng đó được gọi là tính đa hình.
- ❑ Các lớp đối tượng khác nhau có thể có cùng tên phương thức.
- ❑ Lợi ích chính của đa hình:
 - Làm đơn giản cho giao diện lập trình.
 - Nó cho phép các quy ước đã thành lập có thể tái sử dụng ở các lớp sau.
 - Tái sử dụng lại với các tên giống nhau

