

Objective-C

To infinity and beyond !

Plan

- Syntaxe et principes de base
- Structure d'une classe
- Nouveautés
- Tour de Xcode

Syntaxe et principes

Bases

- Objective-C :
- Surcouche du C
- Héritage simple
- Runtime dynamique
- On parle d'envoi de message, et non pas d'appel de méthode

Bases

- Objective-C :
 - Primitifs issus du C (int, float, char, double, etc.)
 - Crées sur la pile
 - Gérés par valeur
 - Objets
 - Crées sur le tas
 - Accès via un pointeur (représenté par le symbole *)
 - Gérés par référence

Syntaxe et principes

C++ :

myObject->myMethod();

Java :

myObject.myMethod();

Syntaxe et principes

C++ :

myObject->myMethod();

Java :

myObject.myMethod();

Objective-C

[myObject myMethod];

Syntaxe et principes

[aCar speed];
objet méthode

[aCar setSpeed:130];
argument

[aCar setSpeed:130 andBrand:aBrand];
méthode argument

Syntaxe

speed

setSpeed:

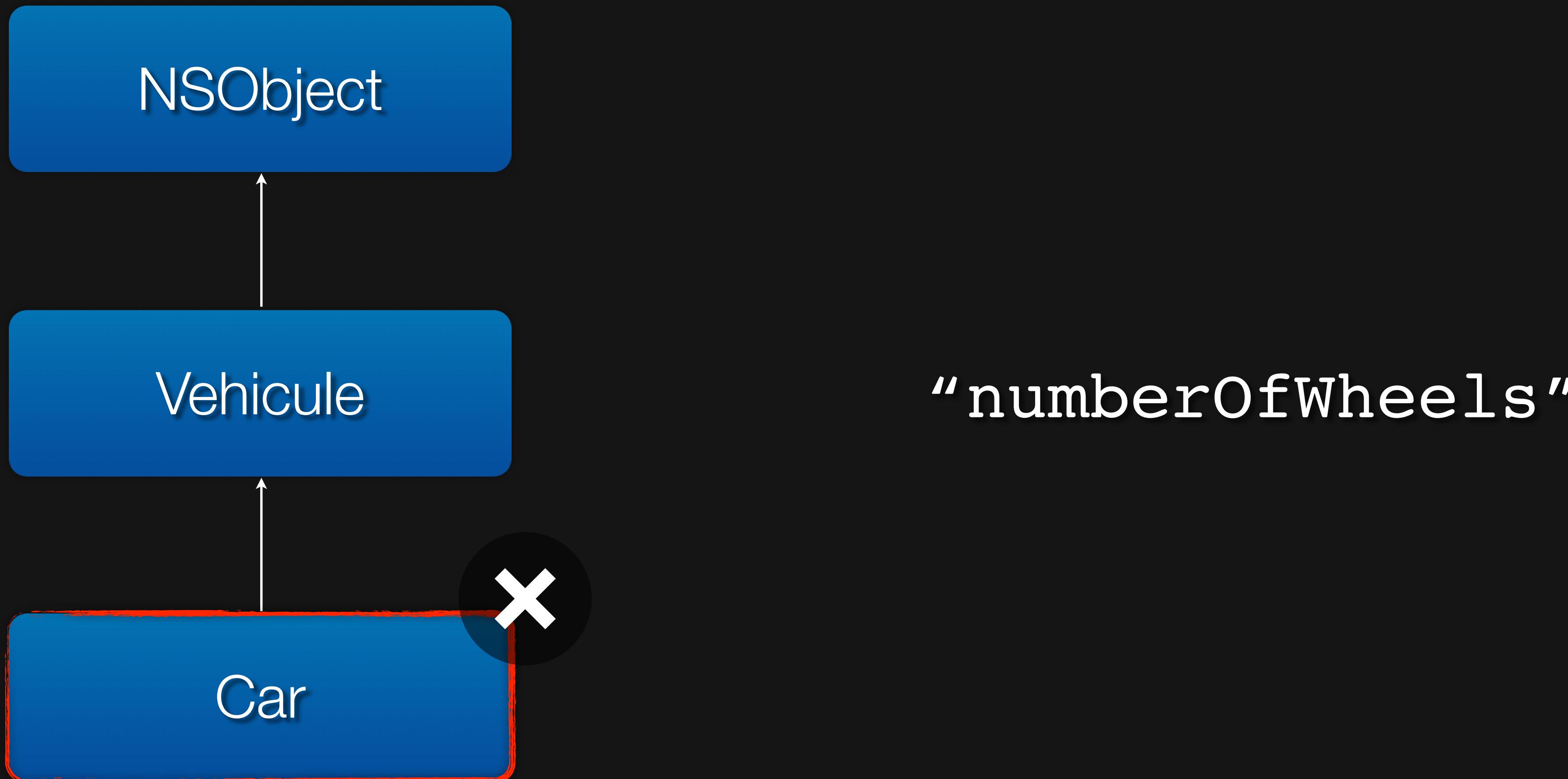
setSpeed:

andBrand:

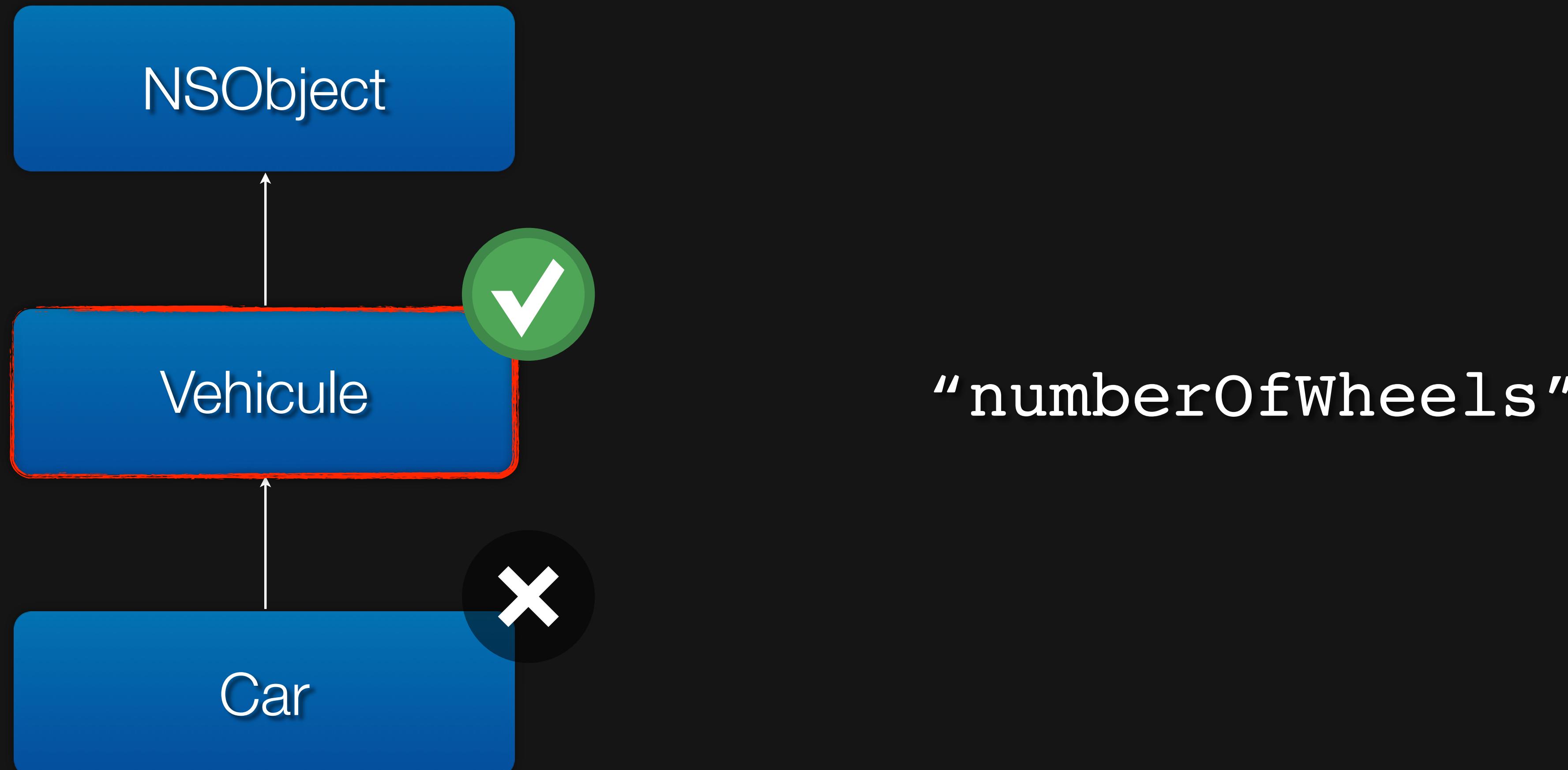
Syntaxe et principes

Message	Sélecteur
[aCar speed]	“speed”
[aCar setSpeed:130];	“setSpeed:”
[aCar setSpeed:130 andBrand:aBrand];	“setSpeed:andBrand:”

Distribution des messages



Distribution des messages



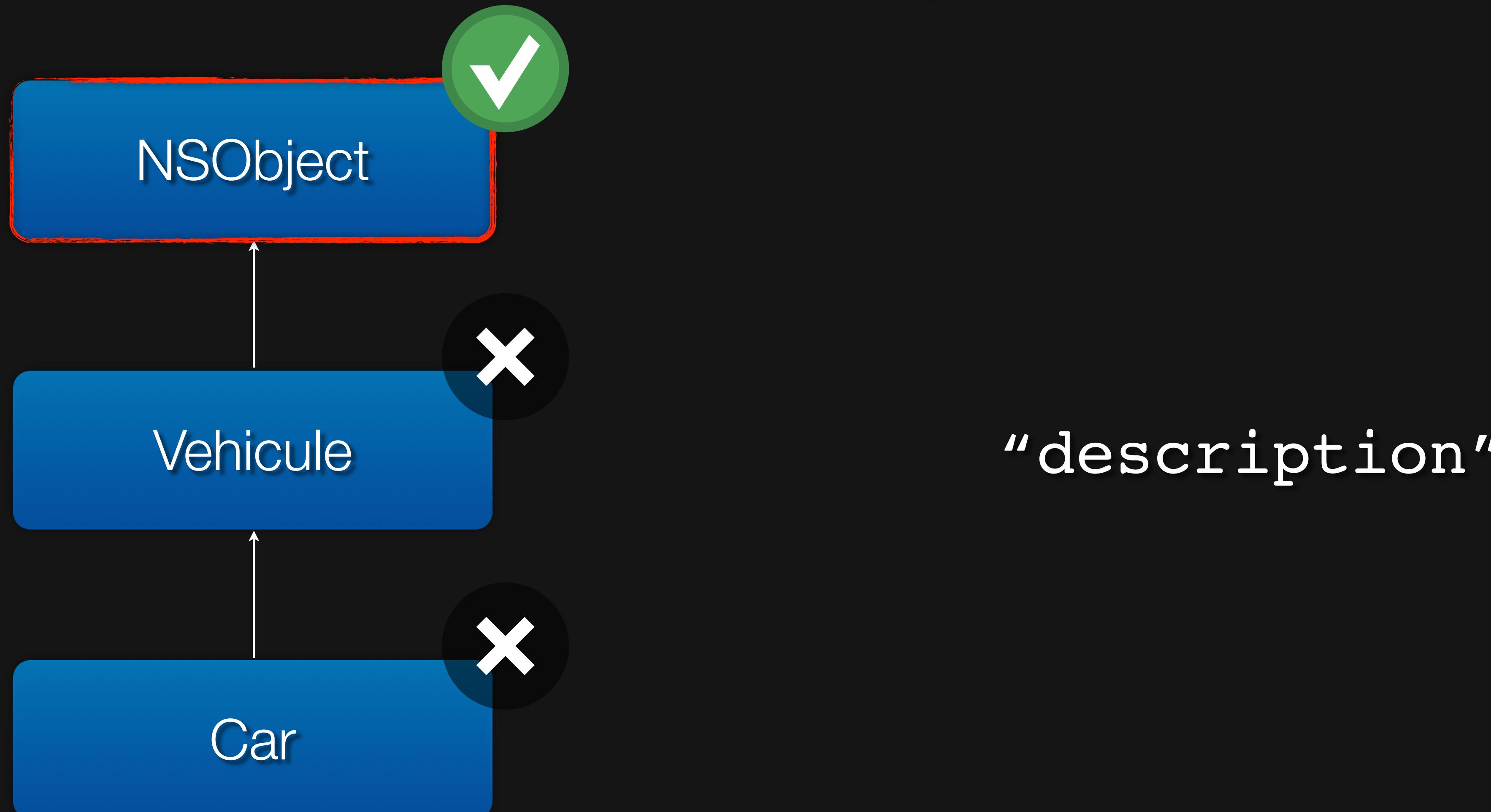
Distribution des messages



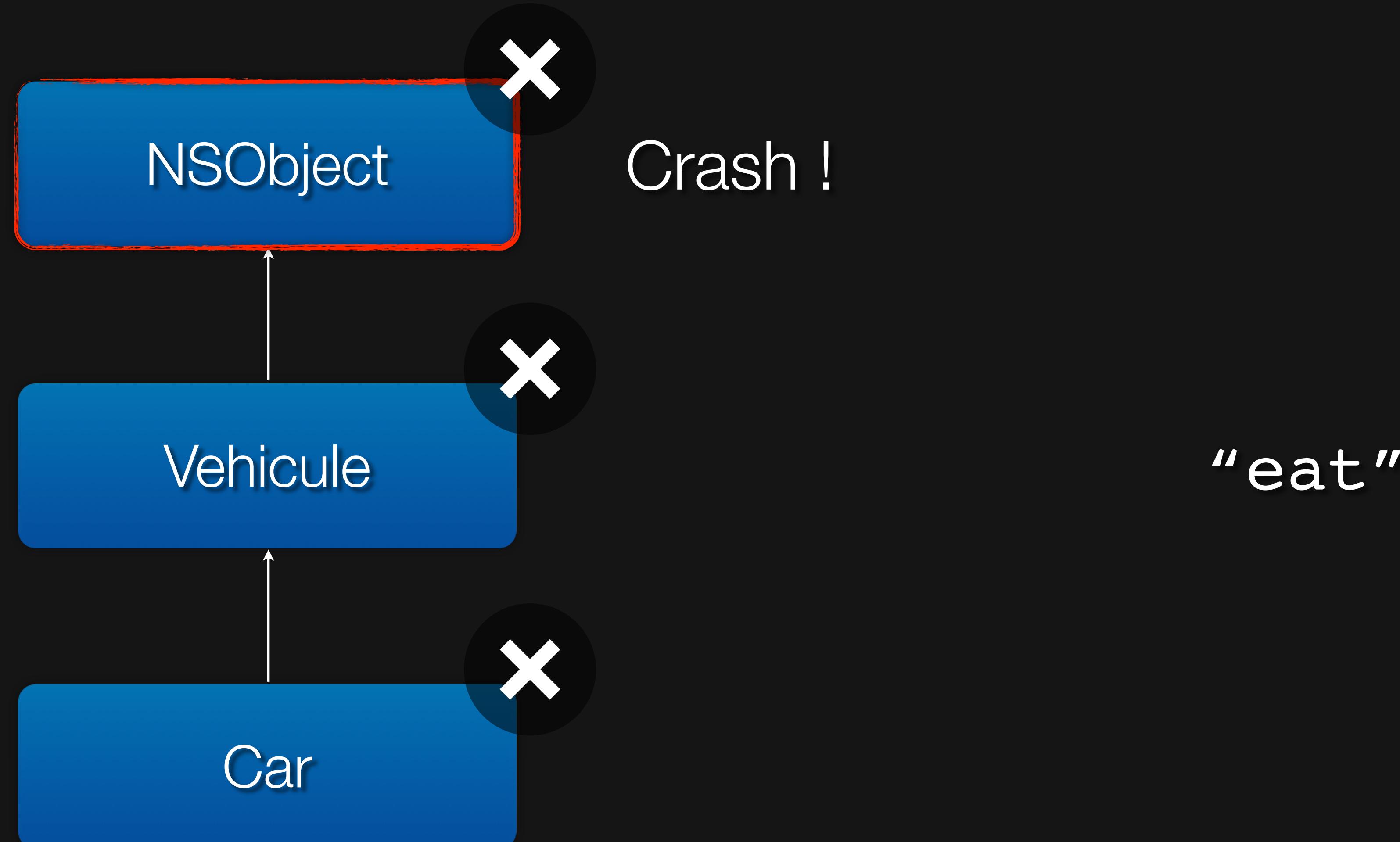
Distribution des messages



Distribution des messages



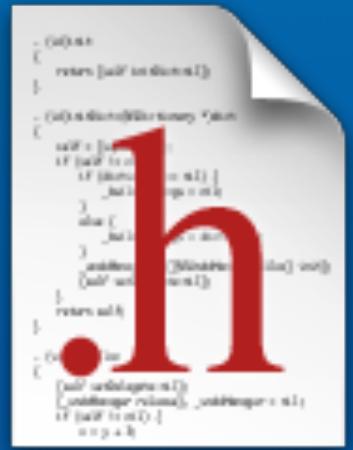
Distribution des messages



*** Terminating app due to uncaught exception 'NSInvalidArgumentException',
reason: '-[Car eat]: unrecognized selector sent to instance 0x7fa18b100070'

Structure d'une classe

Classe Objective-C



Interface



Implémentation

Interface



Interface



- *Fiche d'identité **publique** de la classe*
- Contient la déclaration des variables d'instance
- Contient la déclaration des méthodes **publiques** destinées à être utilisées par d'autres objets

Interface



- Nom de la classe
- Ancêtre (héritage)
- Propriétés (variables d'instance)
- Méthodes
 - de classe
 - d'instance

Structure d'une classe

Interface

```
@interface Car : NSObject
```

```
@property (strong, nonatomic) NSString *brand;  
@property (nonatomic) int speed;
```

```
// Déclaration des méthodes  
@end
```



Propriétés



- Utilisées pour déclarer les variables d'instances
- Déclare une variable
- Génère les accesseurs selon les règles indiquées



Propriétés

Défini comment générer les accesseurs

```
@property (strong, nonatomic) NSString* brand;
```

Est équivalent aux déclarations (et implémentations) suivantes :

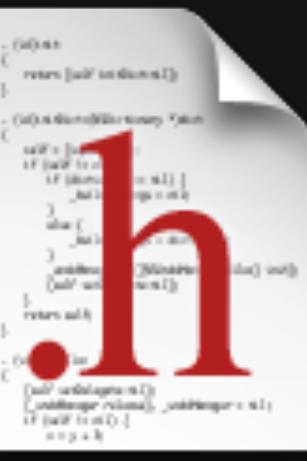
```
NSString * _brand;
```

```
-(void)setBrand:(NSString *)brand;
```

```
-(NSString *)brand;
```

Propriétés

- Gestion mémoire
 - **strong** (défaut pour objet)
 - **weak**
 - **assign** (par défaut pour primitives)
 - **nonatomic**
 - **atomic** (par défaut)
- Multithread
 - **nonatomic**
- Modification
 - **readwrite** (par défaut)
 - **readonly**
 - Nom des méthodes
 - **getter=**
 - **setter=**

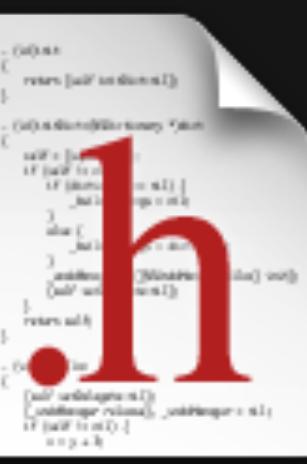


Méthodes

- Type de méthode
- Classe : +
- Instance : -
- Type de retour
- Nom de la méthode et arguments typés



Méthodes d'instance



[aCar speed];

- (int) speed;

[aCar setSpeed:130];

- (void) setSpeed:(int) aSpeed;

[aCar setSpeed:130 andBrand:aBrand];

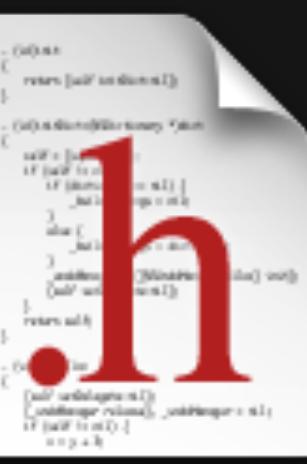
- (void) setSpeed:(int) aSpeed andBrand:(NSString*) aBrand;

Méthodes de classe

```
+ (int) defaultSpeed;
```



Interface



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

@interface Car : NSObject

@property (strong, nonatomic) NSString *brand;
@property (nonatomic) int speed;

+ (int)defaultSpeed;
- (void)setSpeed:(int)aSpeed andBrand:(NSString *)aBrand;
- (void)increaseSpeedOf:(int)val;

- (int)speed;
- (NSString *)brand;
- (void)setSpeed:(int)aSpeed;
- (void)setBrand:(NSString *)aBrand;

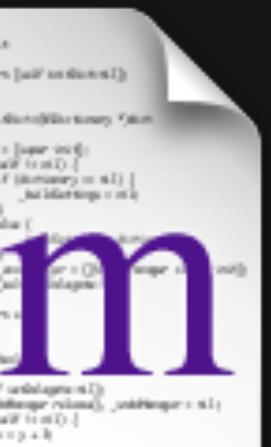
@end
```

Implémentation

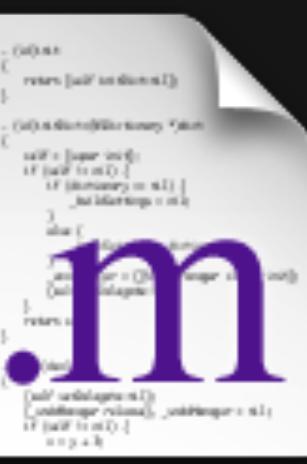
```
- calculate()
{
    return [self calculate];
}

-(id) calculateMemory *item
{
    id = [item memory];
    if (memory > 100)
        _item = [item calculate];
    else
        _item = [item calculate];
    return _item;
}

-(id) calculateWeight *item
{
    id = [item weight];
    if (weight > 100)
        _item = [item calculate];
    else
        _item = [item calculate];
    return _item;
}
```



Implémentation



- *Moteur de la classe*
- Contient le code des méthodes
- Le fichier .m contient parfois une catégorie anonyme, ou extension d'interface
- Permet de déclarer des méthodes et propriétés "privées"

Implémentation

- Nom de la classe
- Implémentation des méthodes



Implémentation

@implementation Car

```
- (void)increaseSpeedOf:(int)val {  
    _speed = _speed + val;  
}
```

@end



Implémentation

```
#import "Car.h"

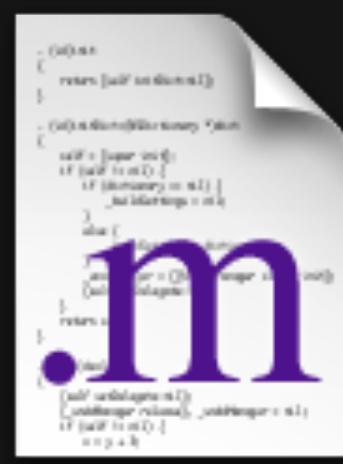
@implementation Car

- (void)increaseSpeedOf:(int)val {
    _speed = _speed + val;
}

@end
```



Extension d'interface



```
#import "Car.h"

@interface Car ()

@property (strong, nonatomic) NSString *aPrivateString;
- (void)aPrivateMethod;

@end

@implementation Car

- (void)increaseSpeedOf:(int)val {
    _speed = _speed + val;
}

@end
```

Initialisation

Initialisation

- Initialiseur
 - Initialise les variables d'instance
 - Il peut en exister plusieurs
- Objets instanciés sur le tas (*heap*), accessibles via un pointeur (*)
- Types primitifs sur la pile (*stack*), accessibles directement

Initialiseur

- Commence toujours par **init**
 - **init**
 - **initWithFrame:**
 - **initWithSpeed:andBrand:**
 - ...

Initialiseur

```
-(instancetype)init {
    self = [super init];
    if (self) {
        // ...
    }
    return self;
}
```

Initialiseur

```
#import "Car.h"

@implementation Car

- (instancetype)init
{
    self = [super init];
    if (self) {

        _brand = @"";

    }
    return self;
}

@end
```

Initialiseur personnalisé

```
#import "Car.h"

@implementation Car

- (instancetype)initWithBrand:(NSString*)brand {

    self = [super init];

    if (self) {

        _brand = brand;
    }

    return self;
}

@end
```

Chaine d'initialiseurs

```
#import "Car.h"

@implementation Car

- (instancetype)initWithBrand:(NSString*)brand {

    self = [super init];

    if (self) {

        _brand = brand;
    }

    return self;
}

@end
```

```
@implementation Car
```

```
- (instancetype)initWithBrand:(NSString*)brand { Designated initialiser
```

```
    self = [super init];
```

```
    if (self) {
```

```
        _brand = brand;
```

```
    }
```

```
    return self;
```

```
}
```

```
- (instancetype)init { Convenience initialiser
```

```
{
```

```
    self = [self initWithBrand:@""];
```

```
    if (self) {
```

```
    }
```

```
    return self;
```

```
}
```

```
@end
```

Créer un objet

- Création en deux temps
 - Allocation mémoire (gérée par NSObject)
 - Initialisation

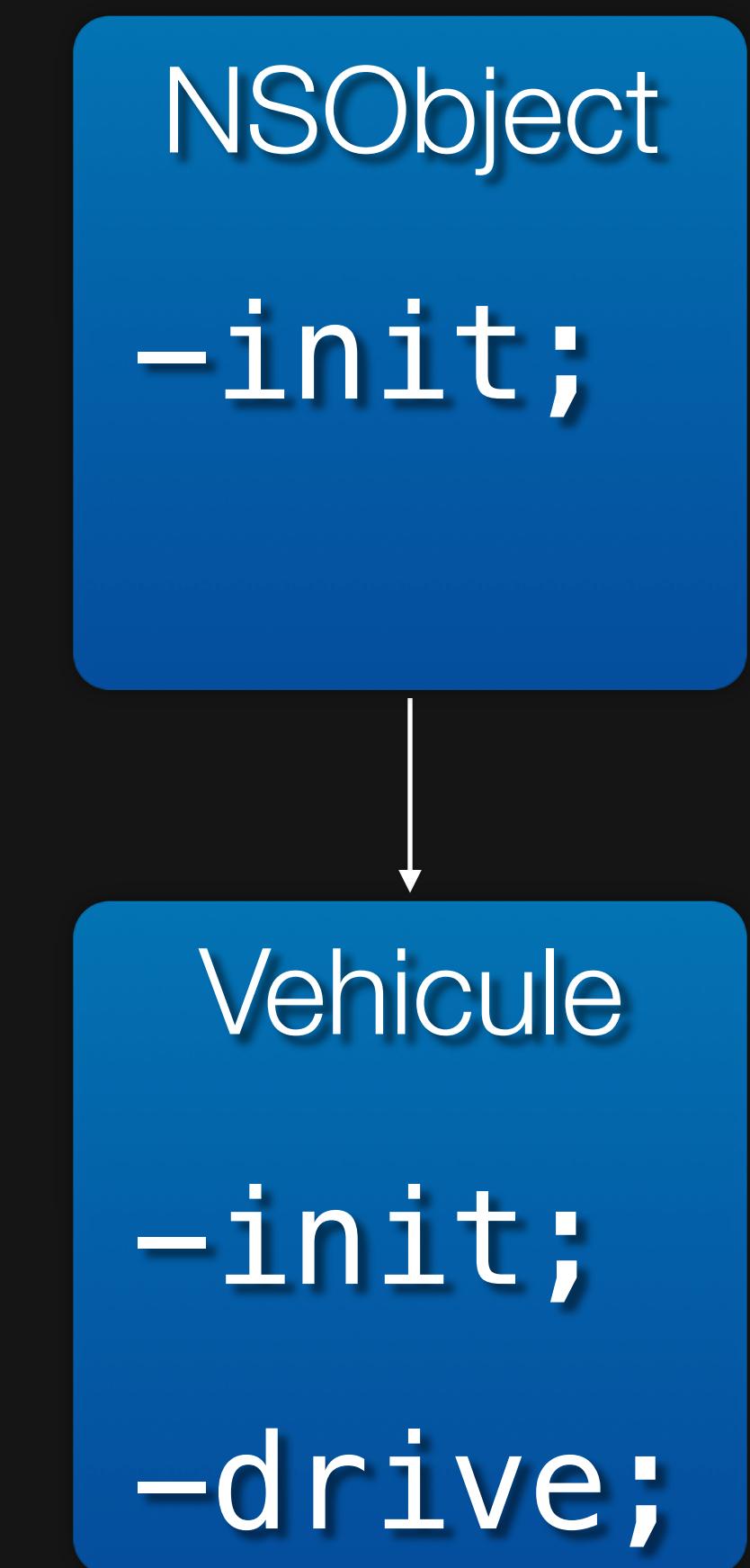
```
Car *aCar = [[Car alloc] init];
Car *anotherCar = [[Car alloc] initWithBrand:@"Audi"];
```

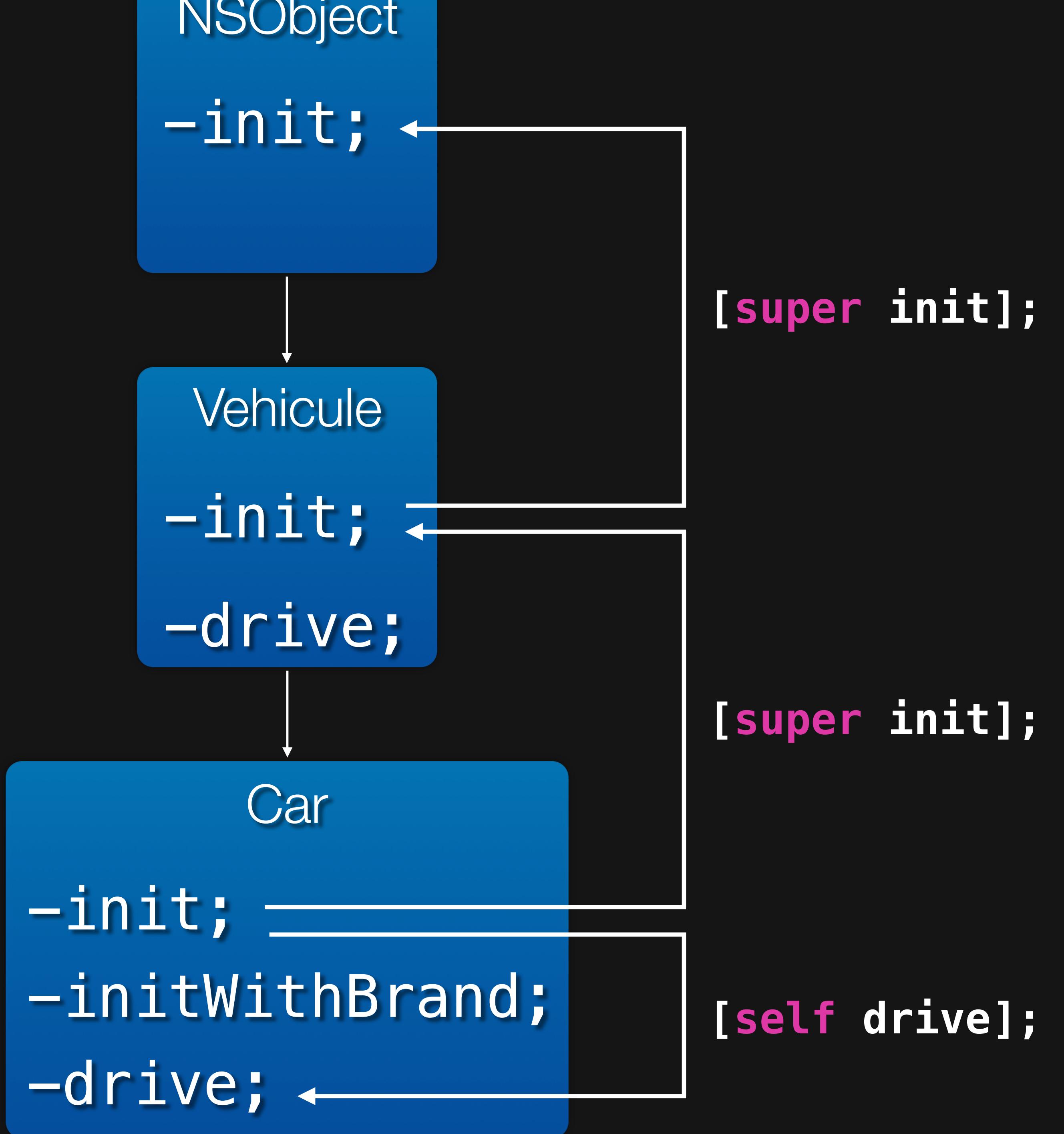
Héritage

Héritage

- Objective-C supporte l'héritage simple
- Classe racine principale : NSObject
 - Nos classes doivent hériter de NSObject pour avoir les mécanismes de base d'instanciation

Héritage





Nouveautés

- Constante

- **nil** : représente un pointeur vers un objet nul

- Un message à **nil** retourne **nil**

- En simplifiant : **nil = NULL = 0**

- Mot clé

- **#import** : équivalent à **#include** mais évite les inclusions multiples

- **@import** peut aussi être utilisé depuis Xcode 5 / iOS 7 pour importer des modules (frameworks)

Nouveautés

- Types :
 - **BOOL**
 - Type booléen
 - 2 valeurs : **YES** et **NO**
 - **NO** = 0 et **YES** = n'importe quoi d'autre
 - **id**
 - Pointeur vers un objet générique

id

```
typedef struct objc_class *Class;  
  
typedef struct objc_object {  
    Class isa;  
} *id;
```

- id : un pointeur vers un objet Objective-C (typage faible)
- La classe d'un objet peut être déterminée à l'exécution (introspection)

Typage dynamique

```
NSString *aString = @“This is a string”; // Typage statique
id obj = aString;                      // Typage dynamique
NSArray *anArray = obj;                 // //!\ Attention
```

- Préciser le type aide à éviter les erreurs
- Le compilateur peut nous prévenir de l’envoi de message “non conforme”

Typage dynamique

✓ `Car *aCar = [[Car alloc] init];`
✓ `[aCar start];`
✓ `[aCar drive];`

✓ `Vehicule *aVehicule = aCar;`
! `[aVehicule openDoor];`

Vehicule

-start;

-drive;

Car

-openDoor;

Typage dynamique

```
✓ Car *aCar = [[Car alloc] init];  
✓ [aCar start];  
✓ [aCar drive];
```

```
✓ id aVehicule = aCar;  
✓ [aVehicule openDoor];
```

Vehicule

-start;

-drive;

Car

-openDoor;

Typage dynamique



```
✓ Car *aCar = [[Car alloc] init];  
✓ [aCar start];  
✓ [aCar drive];
```



```
✓ id aVehicule = aCar;  
! [aVehicule aMethodThatNobodyRespondsTo];
```

Vehicule

-start;

-drive;

Car

-openDoor;

Typage dynamique

```
✓ Car *aCar = [[Car alloc] init];  
✓ [aCar start];  
✓ [aCar drive];
```

```
✓ id aVehicule = aCar;  
✓ [aVehicule bark];
```

Dog

-bark;

Vehicule

-start;

-drive;

Car

-openDoor;

Introspection

- Utiliser id peut être dangereux, mais est parfois utile
 - Cas d'un tableau qui contient des objets de types divers
 - Besoin de faire le point sur le type avant d'exécuter une méthode
 - Utilisation de l'introspection !

Introspection

```
Car *aCar = [[Car alloc] init];
[aCar start];
[aCar drive];

id aVehicule = aCar;

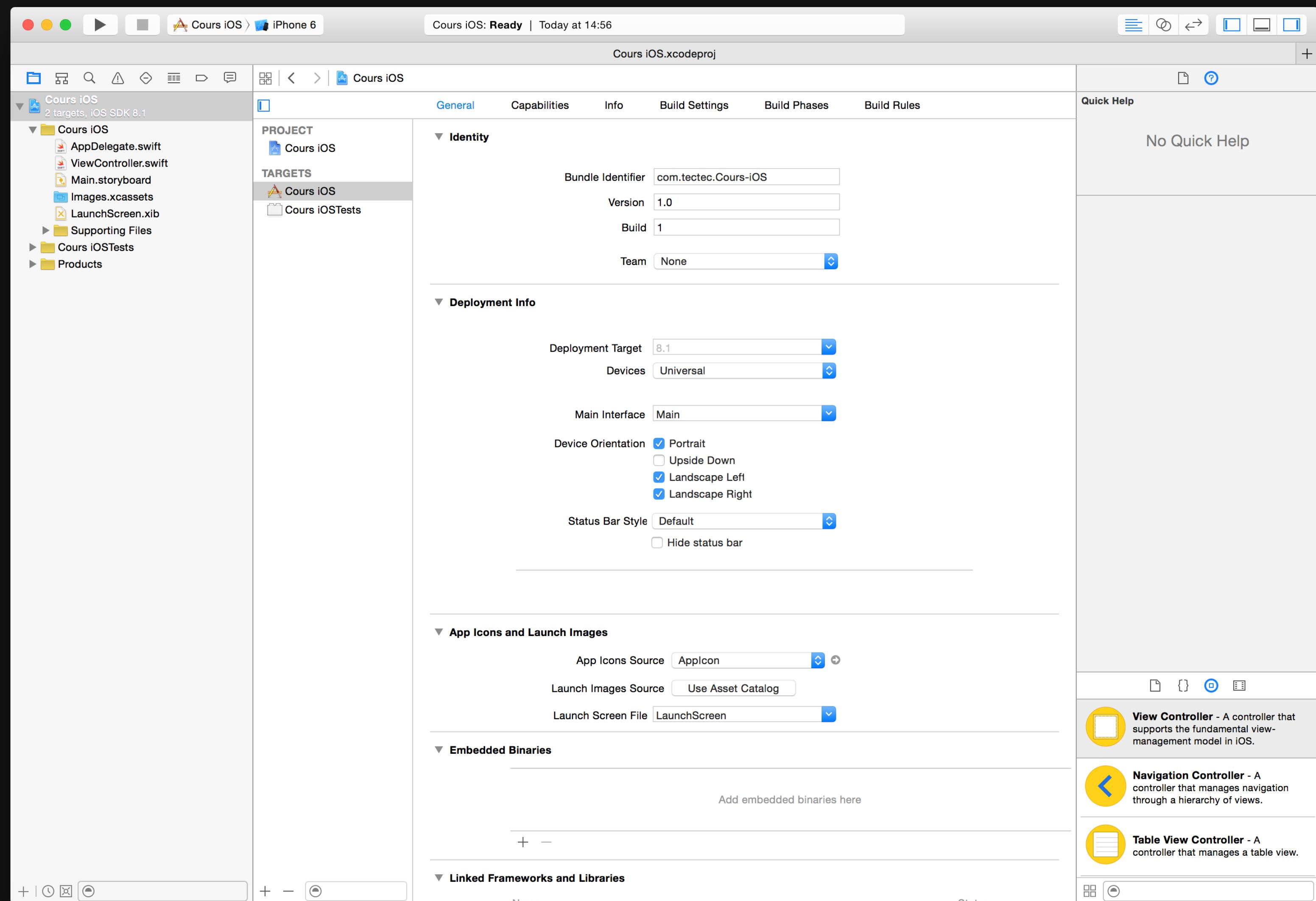
if ([aVehicule isKindOfClass:[Dog class]])
{
    [aVehicule bark];
}
```

Xcode 7

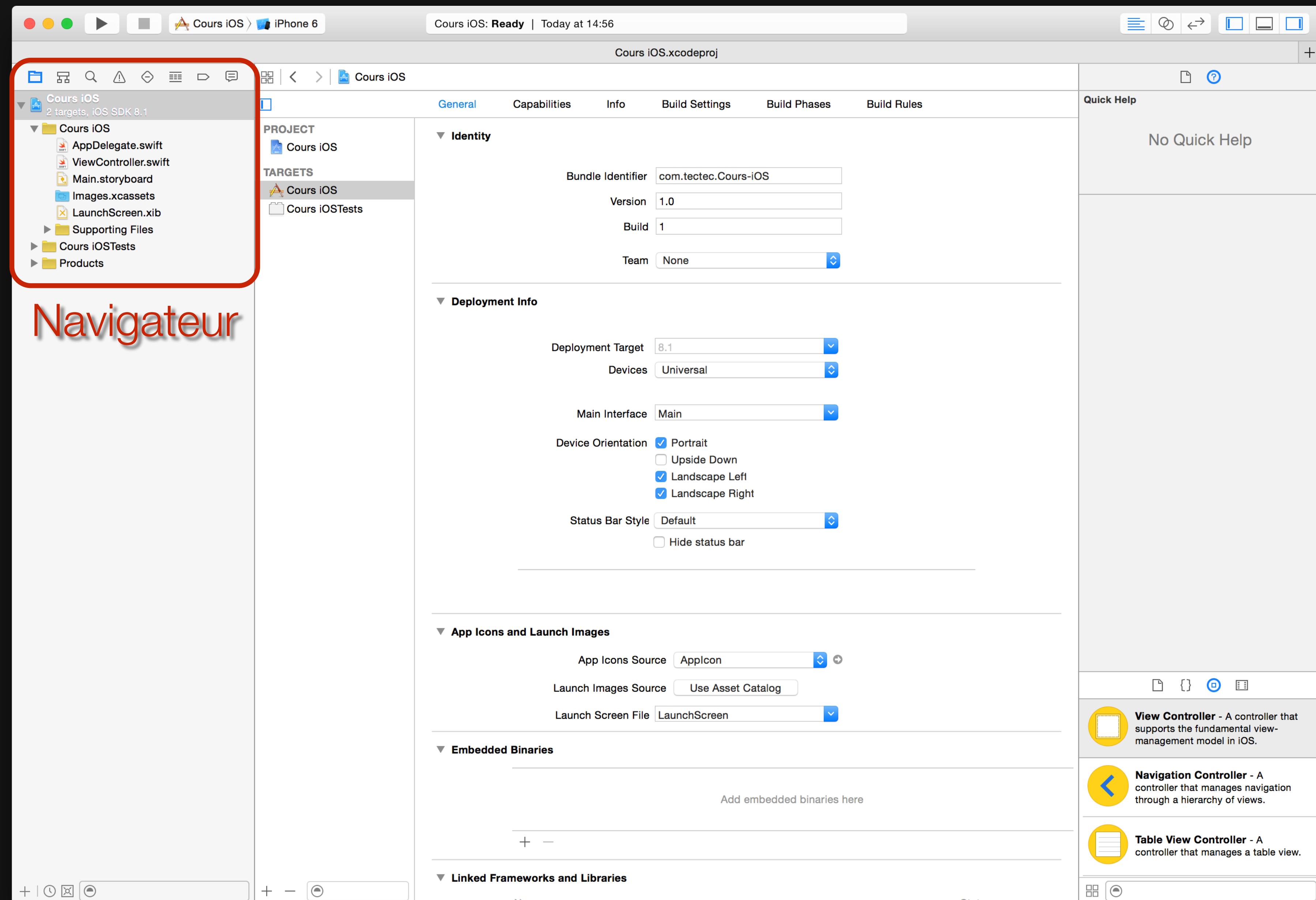
- Xcode 7 a rajouté des fonctionnalités à Objective-C
 - Principalement pour améliorer la compatibilité avec Swift
 - Renforce également la robustesse du code Objective-C
- Annotations de nullabilité
- Génériques légers

Tour de Xcode

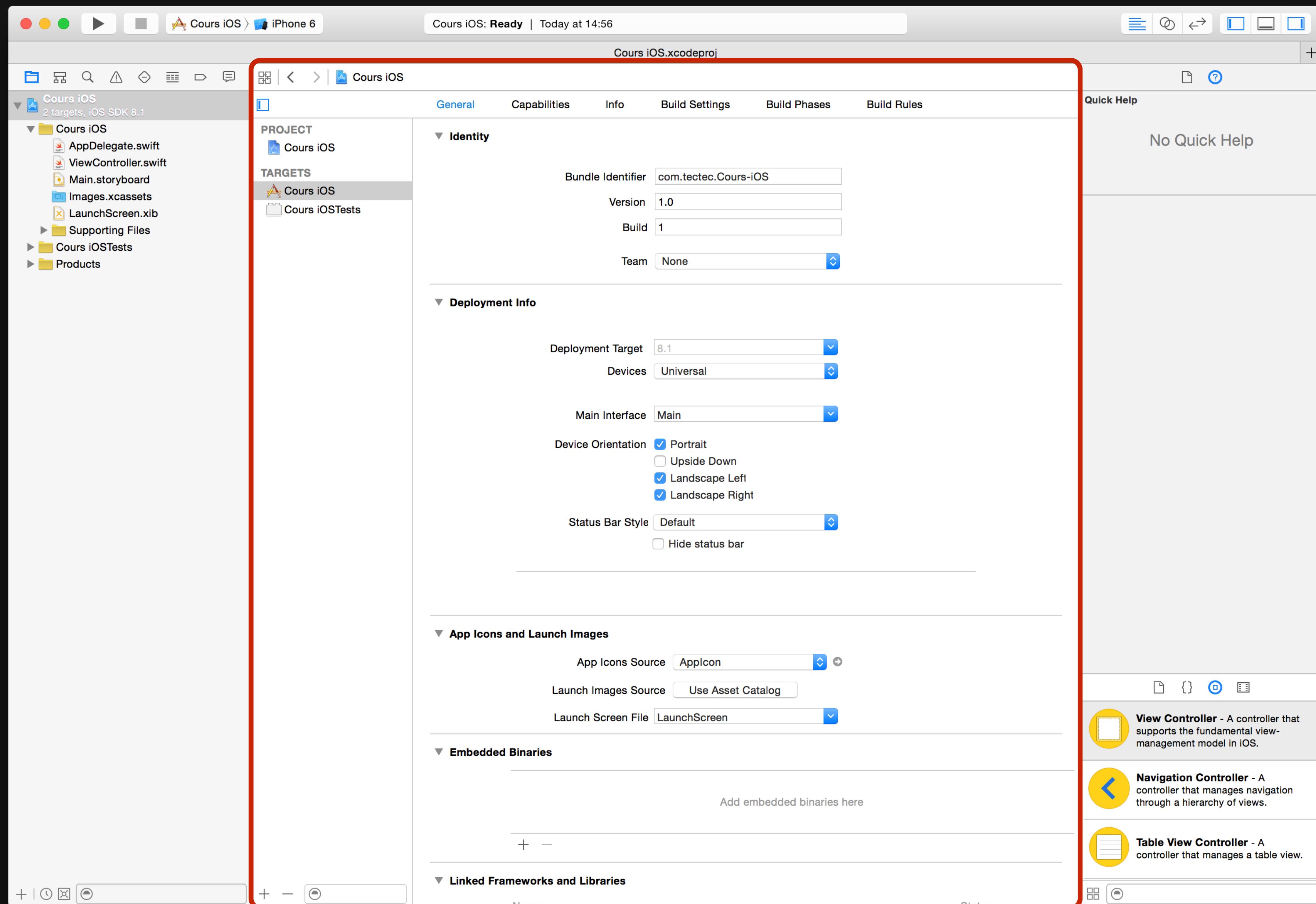
Architecture d'un projet Xcode



Tour de Xcode



Tour de Xcode



Vue centrale

Tour de Xcode

Cours iOS: Ready | Today at 14:56

Cours iOS.xcodeproj

Quick Help
No Quick Help

Inspecteur

PROJECT
Cours iOS

TARGETS
Cours iOS

Identity

- Bundle Identifier: com.tectec.Cours-iOS
- Version: 1.0
- Build: 1
- Team: None

Deployment Info

- Deployment Target: 8.1
- Devices: Universal
- Main Interface: Main
- Device Orientation:
 - Portrait (checked)
 - Upside Down
 - Landscape Left (checked)
 - Landscape Right (checked)
- Status Bar Style: Default
- Hide status bar

App Icons and Launch Images

- App Icons Source: AppIcon
- Launch Images Source: Use Asset Catalog
- Launch Screen File: LaunchScreen

Embedded Binaries

Add embedded binaries here

Linked Frameworks and Libraries

Name Status

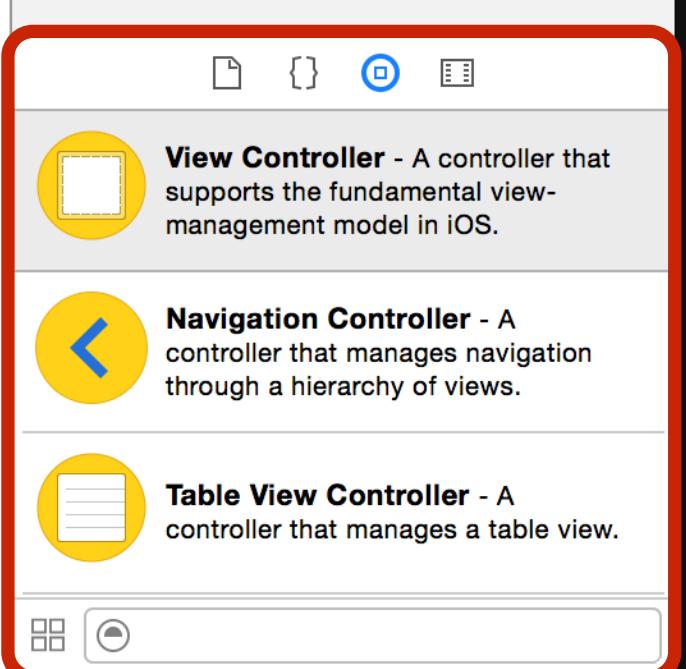
View Controller - A controller that supports the fundamental view-management model in iOS.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

Table View Controller - A controller that manages a table view.

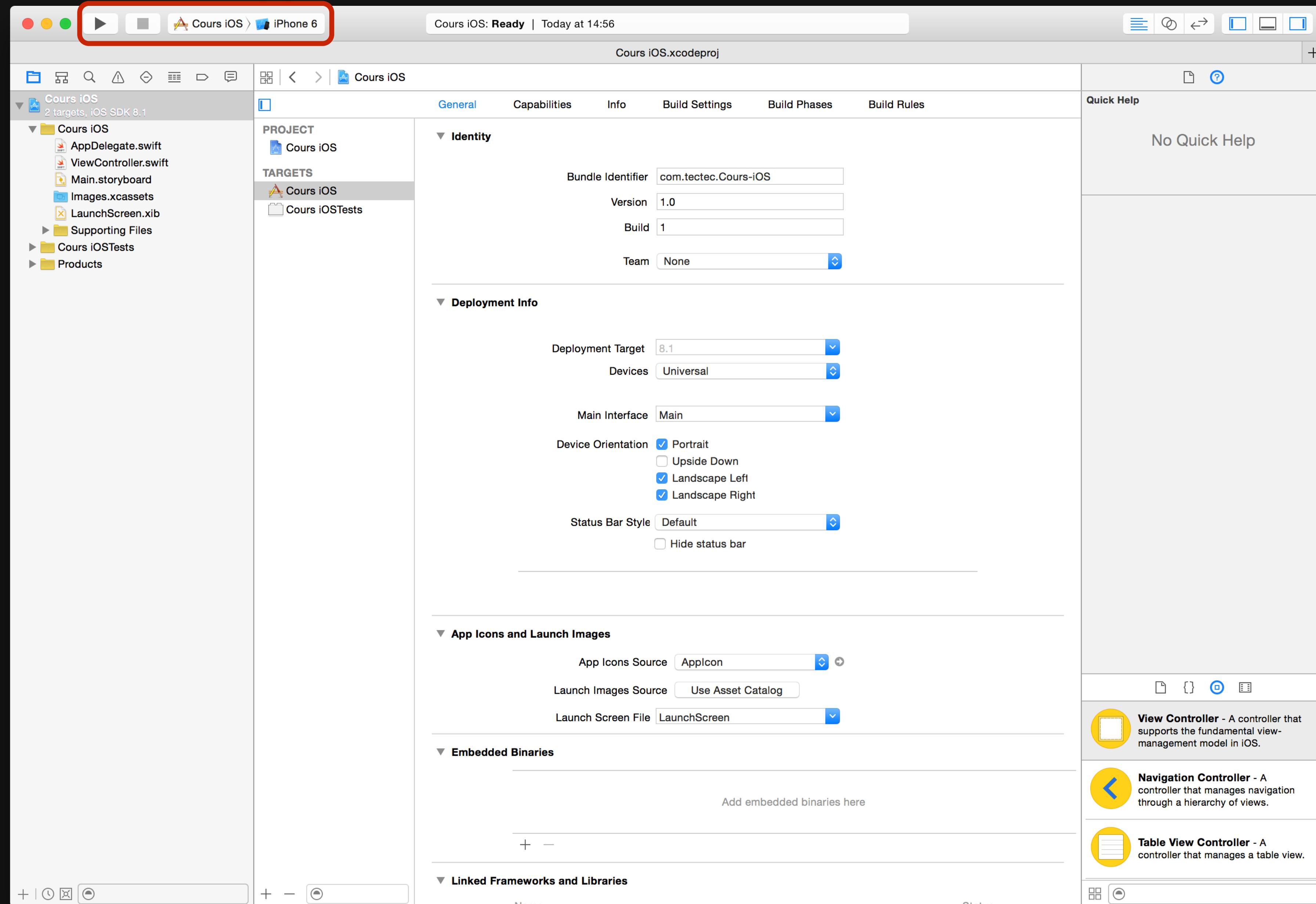
Tour de Xcode

The screenshot shows the Xcode interface with the project 'Cours iOS' selected. The left sidebar lists files like AppDelegate.swift, ViewController.swift, Main.storyboard, etc. The main area shows the 'General' tab of the project settings. Under 'Identity', the bundle identifier is set to 'com.tectec.Cours-iOS'. Under 'Deployment Info', the deployment target is '8.1', devices are 'Universal', and the main interface is 'Main'. Under 'App Icons and Launch Images', the app icons source is 'AppIcon' and the launch images source is 'Use Asset Catalog'. Under 'Embedded Binaries', there is a placeholder for adding embedded binaries. At the bottom, there is a section for 'Linked Frameworks and Libraries'.



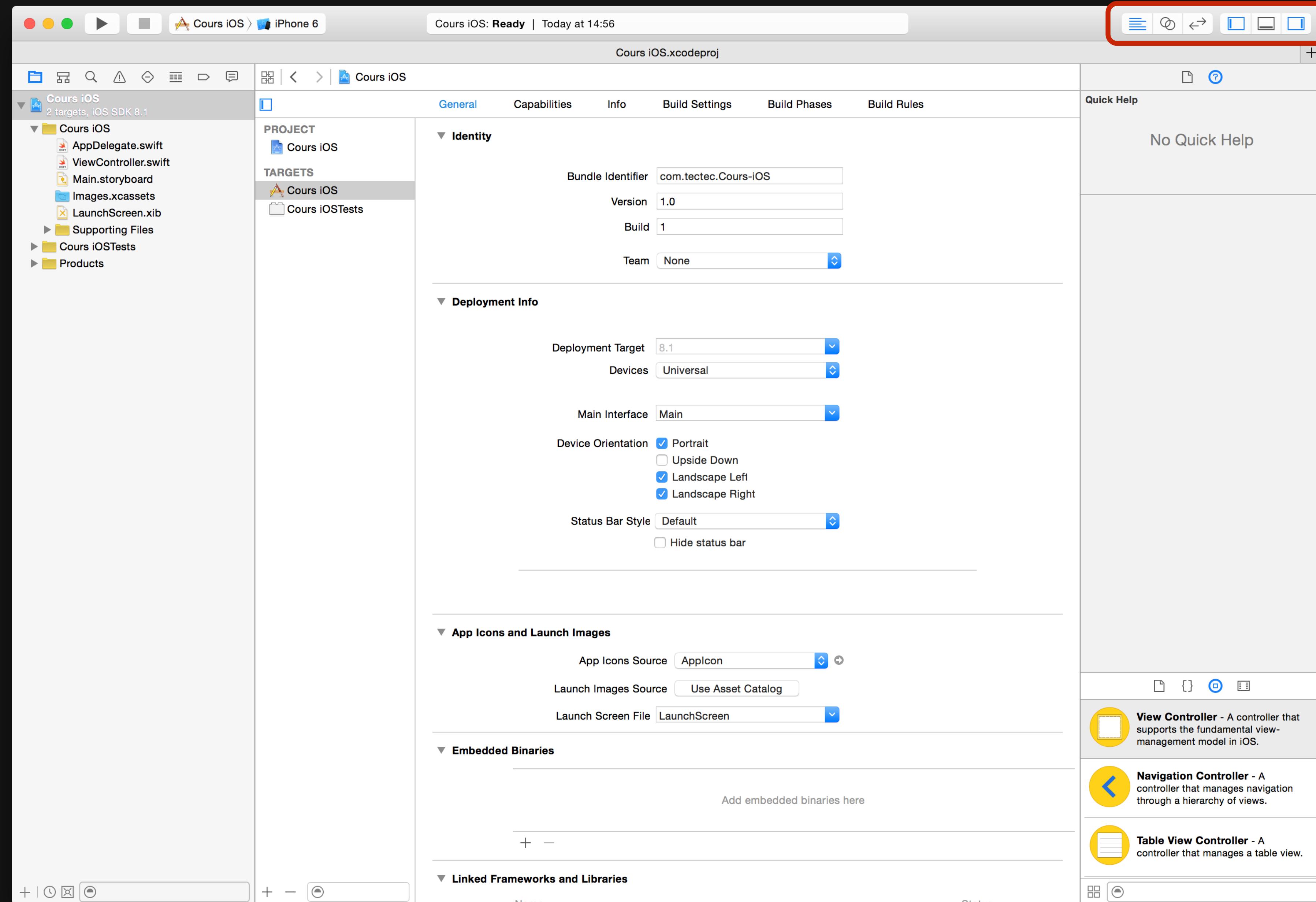
Bibliothèque

Tour de Xcode Contrôle de l'application



Tour de Xcode

Contrôle des vues



Architecture d'un projet Xcode

- Fichiers .h/.m
 - Code
- AppDelegate permet de réagir aux changements d'états de l'application
- Fichiers .storyboard
 - Interface graphique
- Fichiers .xcassets
 - Bibliothèque de ressources graphiques

Pour aller plus loin . . .

- <http://developer.apple.com>
- [The Objective-C Programming Language](#)

