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.)
    - Créés sur la pile
    - Gérés par valeur
  - Objets
    - Créés sur le tas
    - Accès via un pointeur (représenté par le symbole *)
    - Gérés par référence

C++ : `myObject->myMethod();`

Java : `myObject.myMethod();`

C++ :
```
myObject->myMethod();
```

Java :
```
myObject.myMethod();
```

Objective-C

```
[myObject myMethod];
```

Syntaxe et principes

```
[aCar speed];
```

```
[aCar speed];
```

objet méthode

```
[aCar speed];
```

objet méthode

```
[aCar setSpeed:130];
```

```
[aCar speed];
```

```
[aCar setSpeed:130];
```

objet méthode argument

```
[aCar speed];
```

```
[aCar setSpeed:130];
```

objet méthode argument

```
[aCar setSpeed:130 andBrand:aBrand];
```

```
[aCar speed];
```

```
[aCar setSpeed:130];
```

```
[aCar setSpeed:130 andBrand:aBrand];
```

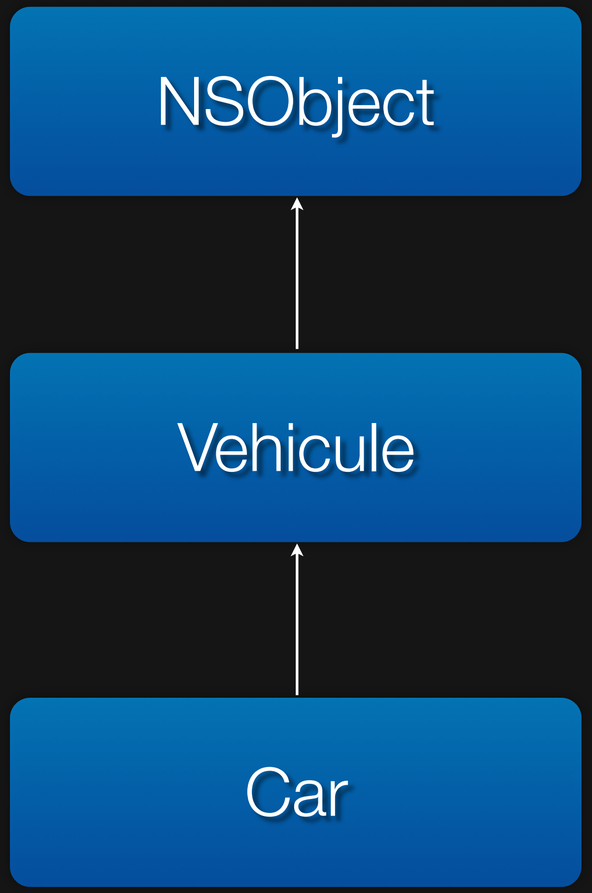
objet méthode argument méthode argument

Syntaxe

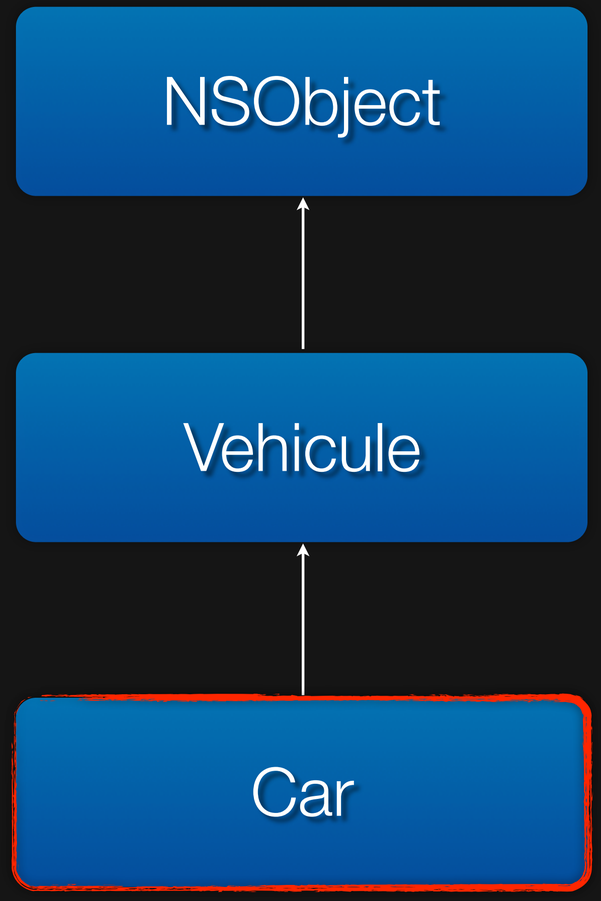
```
speed

setSpeed:

setSpeed:    andBrand:
```

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

Distribution des messages

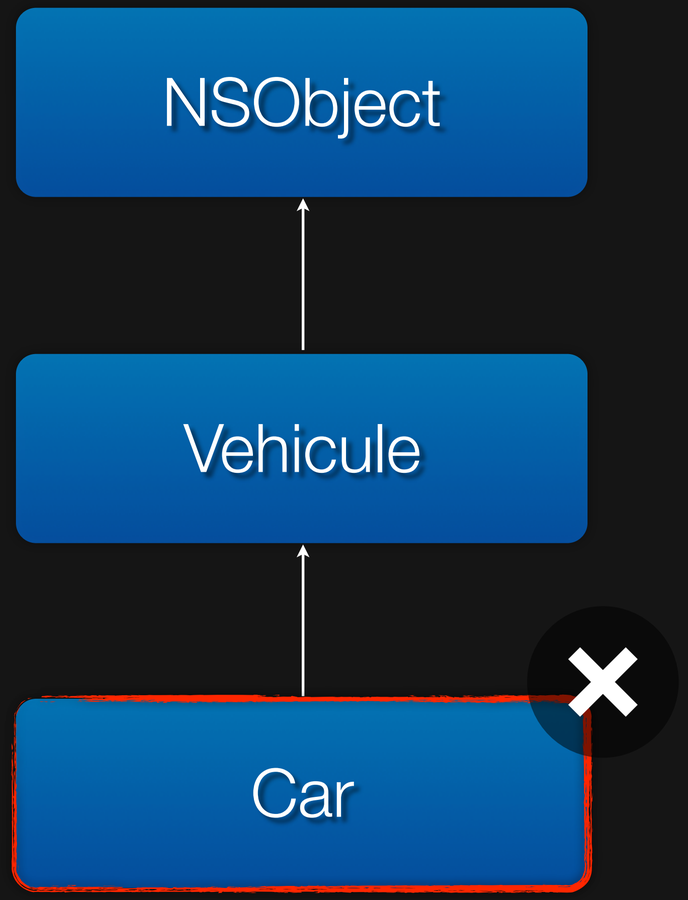


Distribution des messages



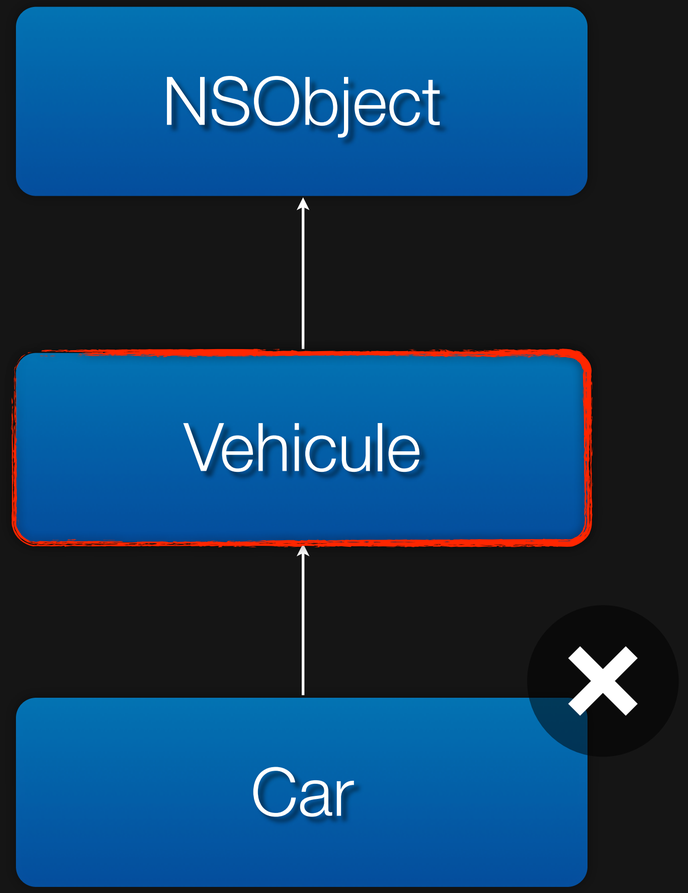

```
“numberOfWheels”
```

Distribution des messages



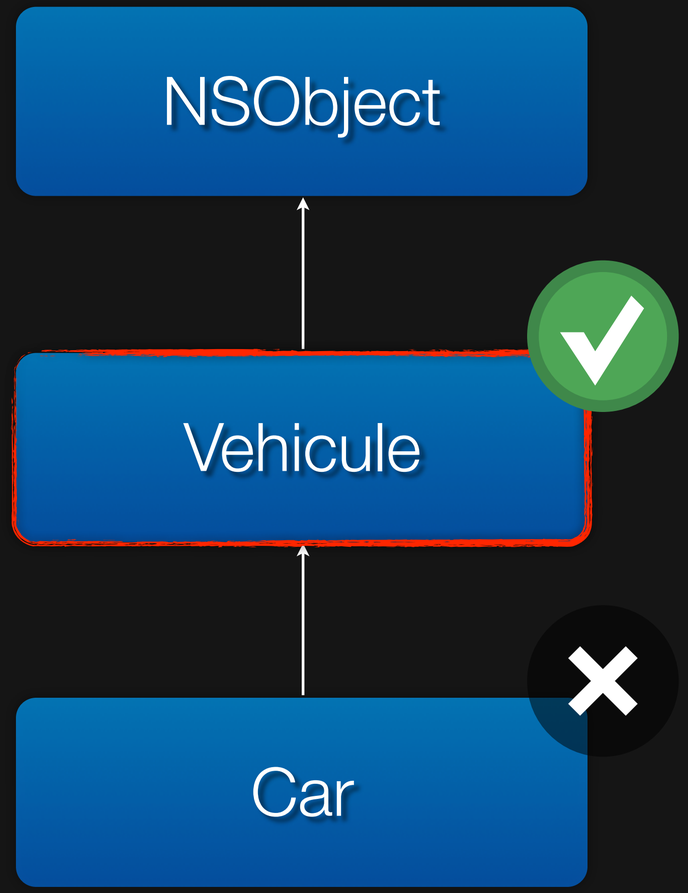

```
“numberOfWheels”
```

Distribution des messages



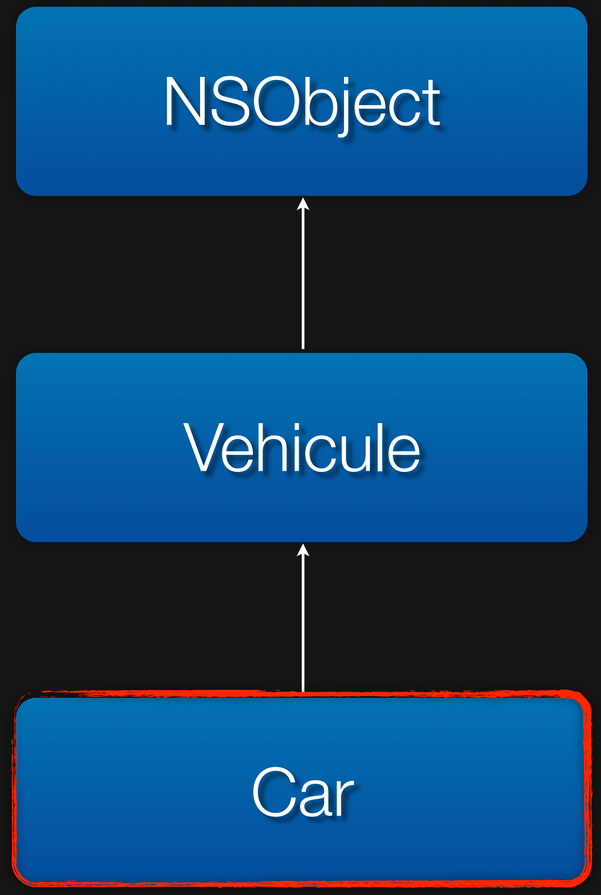

```
“numberOfWheels”
```

Distribution des messages



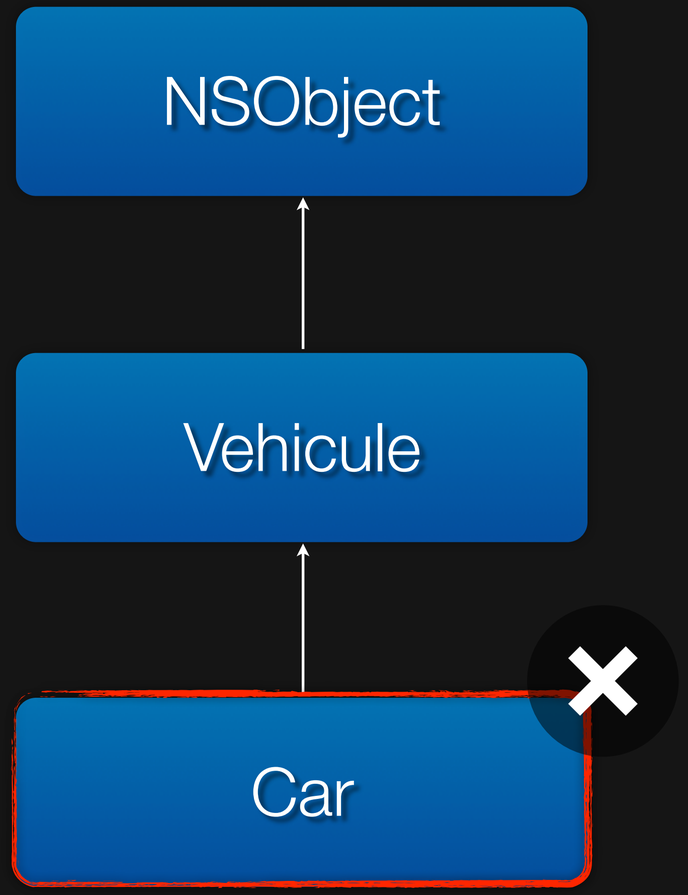
`“numberOfWheels”`

Distribution des messages



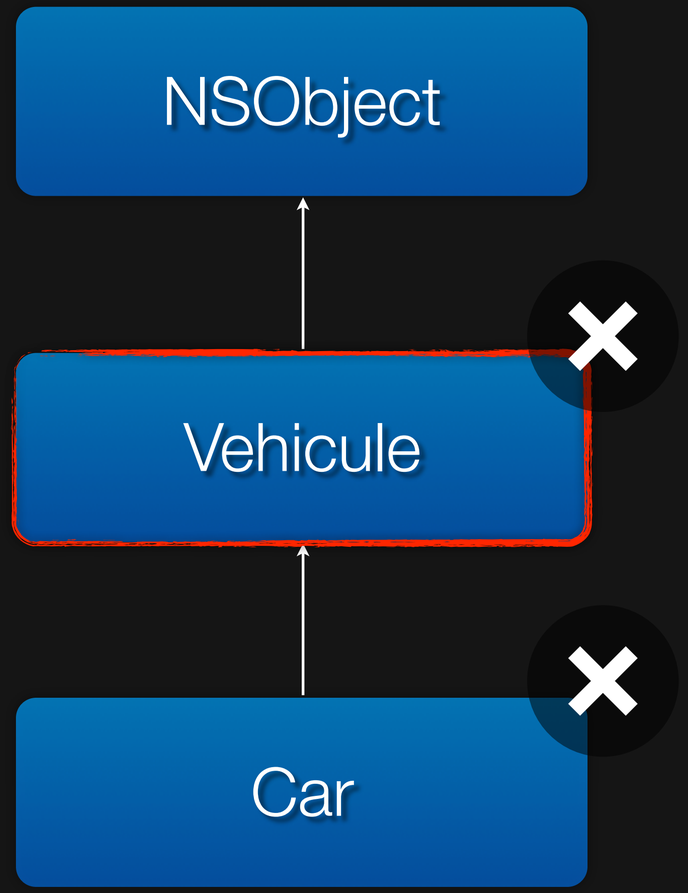
“description”

Distribution des messages



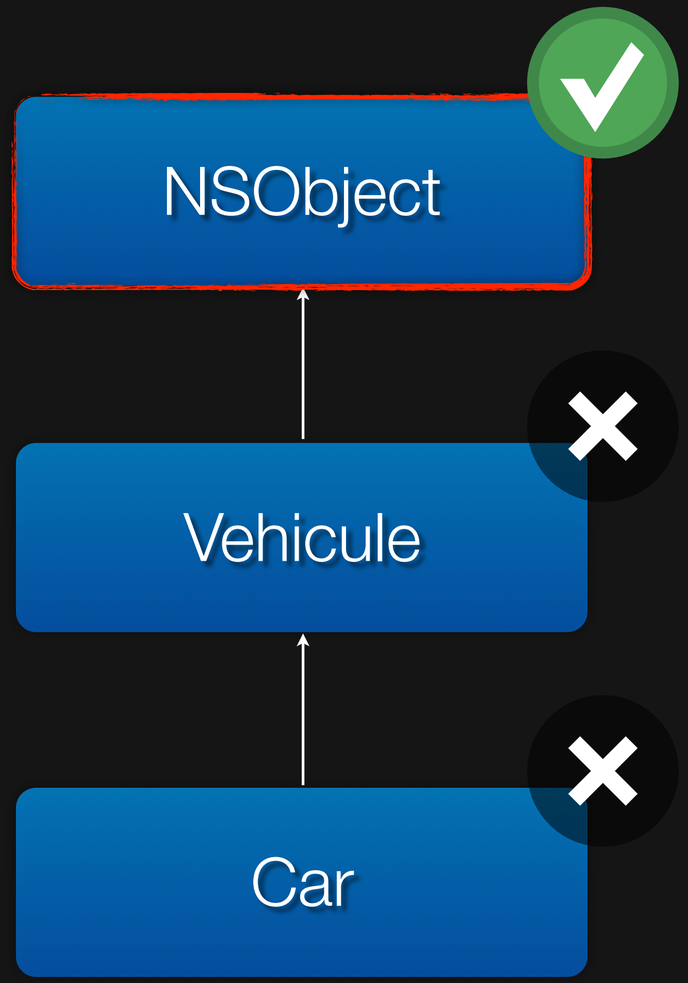

```
“description”
```

Distribution des messages



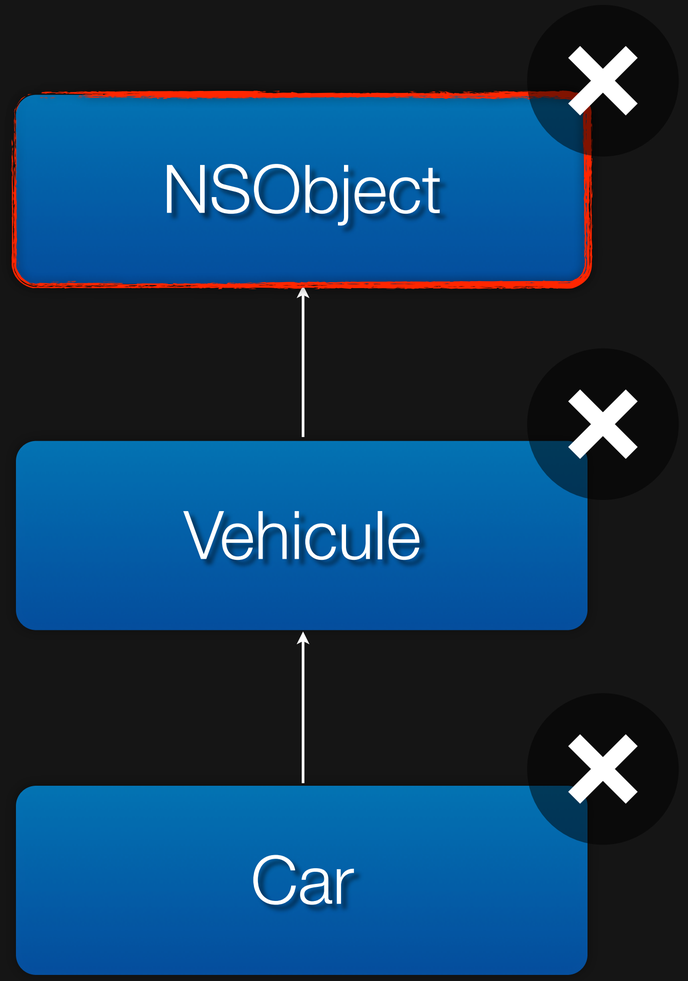

```
“description”
```

Distribution des messages



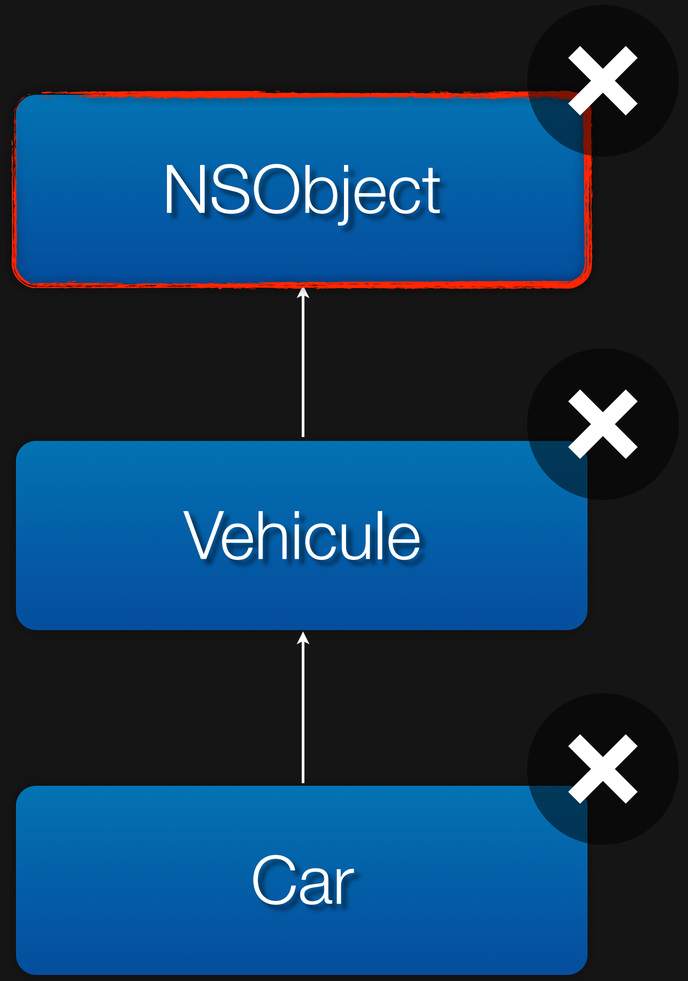

```
“description”
```

Distribution des messages



`“eat”`

Distribution des messages



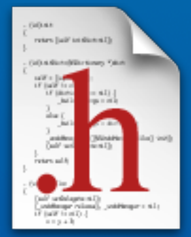
Crash !

“eat”

```
*** 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

Interface

Interface

```
@interface



@end
```

Interface

```
@interface Car



@end
```

Interface

```
@interface Car : NSObject



@end
```

Interface


```
@interface Car : NSObject

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


@end
```

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 primitifs)`**
- Multithread
  - **`nonatomic`**
  - **`atomic (par défaut)`**
- 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];
```

```
[aCar setSpeed:130];
```

```
[aCar setSpeed:130 andBrand:aBrand];
```

Méthodes d’instance

```
[aCar speed];
- (int)speed;

[aCar setSpeed:130];


[aCar setSpeed:130 andBrand:aBrand];
```

Méthodes d’instance

```
[aCar speed];
- (int)speed;

[aCar setSpeed:130];
- (void)setSpeed:(int)aSpeed;

[aCar setSpeed:130 andBrand:aBrand];
```

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 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

Méthodes de classe

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

Interface

```objc
#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
```

Interface


```objc
#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;




@end
```

Implémentation

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

Implémentation


```
@implementation



@end
```

Implémentation

```
@implementation Car



@end
```

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
```

```objc
@implementation Car

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

    self = [super init];

    if (self) {

        _brand = brand;
    }
   return self;
}

- (instancetype)init
{
    self = [self initWithBrand:@""];
    if (self) {

    }
    return self;
}
@end
```

```objc
@implementation Car

- (instancetype)initWithBrand:(NSString*)brand {
    self = [super init];

    if (self) {

        _brand = brand;
    }
   return self;
}

- (instancetype)init
{
    self = [self initWithBrand:@""];
    if (self) {

    }
    return self;
}
@end
```

Designated initialiser

```objc
@implementation Car

- (instancetype)initWithBrand:(NSString*)brand {
    self = [super init];

    if (self) {

        _brand = brand;
    }
   return self;
}

- (instancetype)init
{
    self = [self initWithBrand:@""];
    if (self) {

    }
    return self;
}
@end
```

Designated initialiser

Convenience initialiser

Créer un objet

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

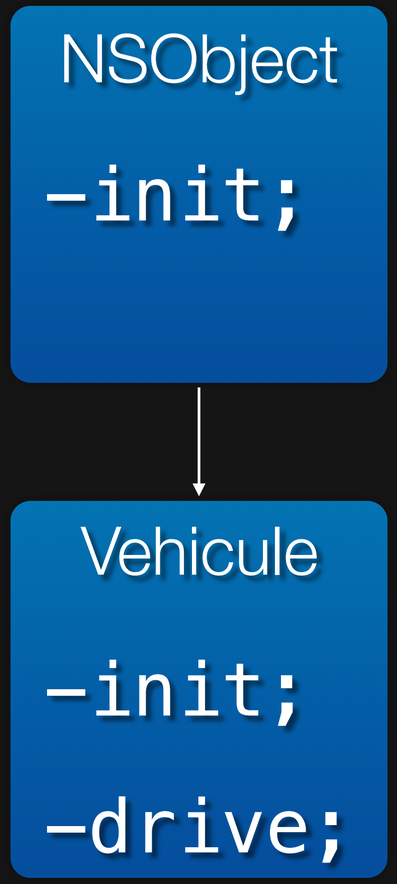
```
Car *aCar = [[Car alloc] init];
Car *anotherCar = [[Car alloc] initWithBrand:@"Audi"];
```

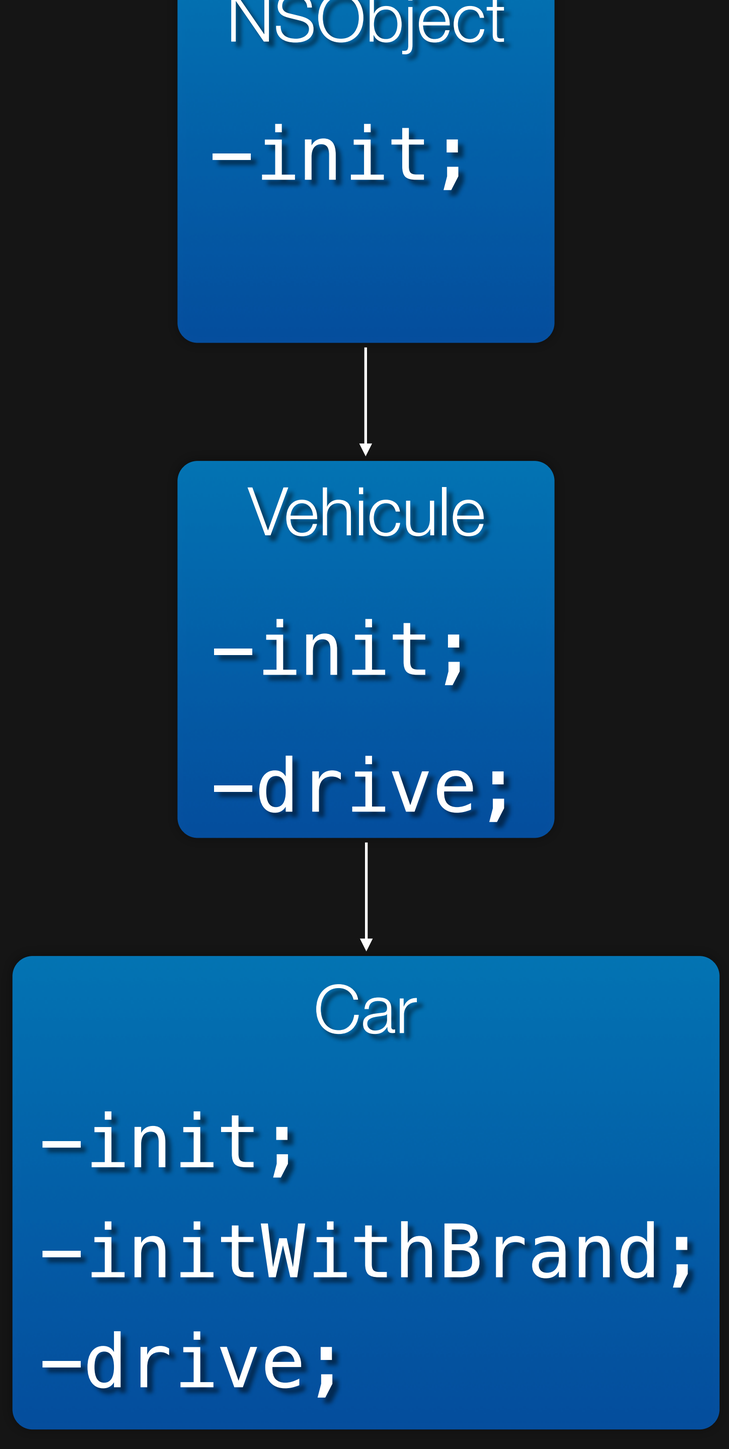
Héritage

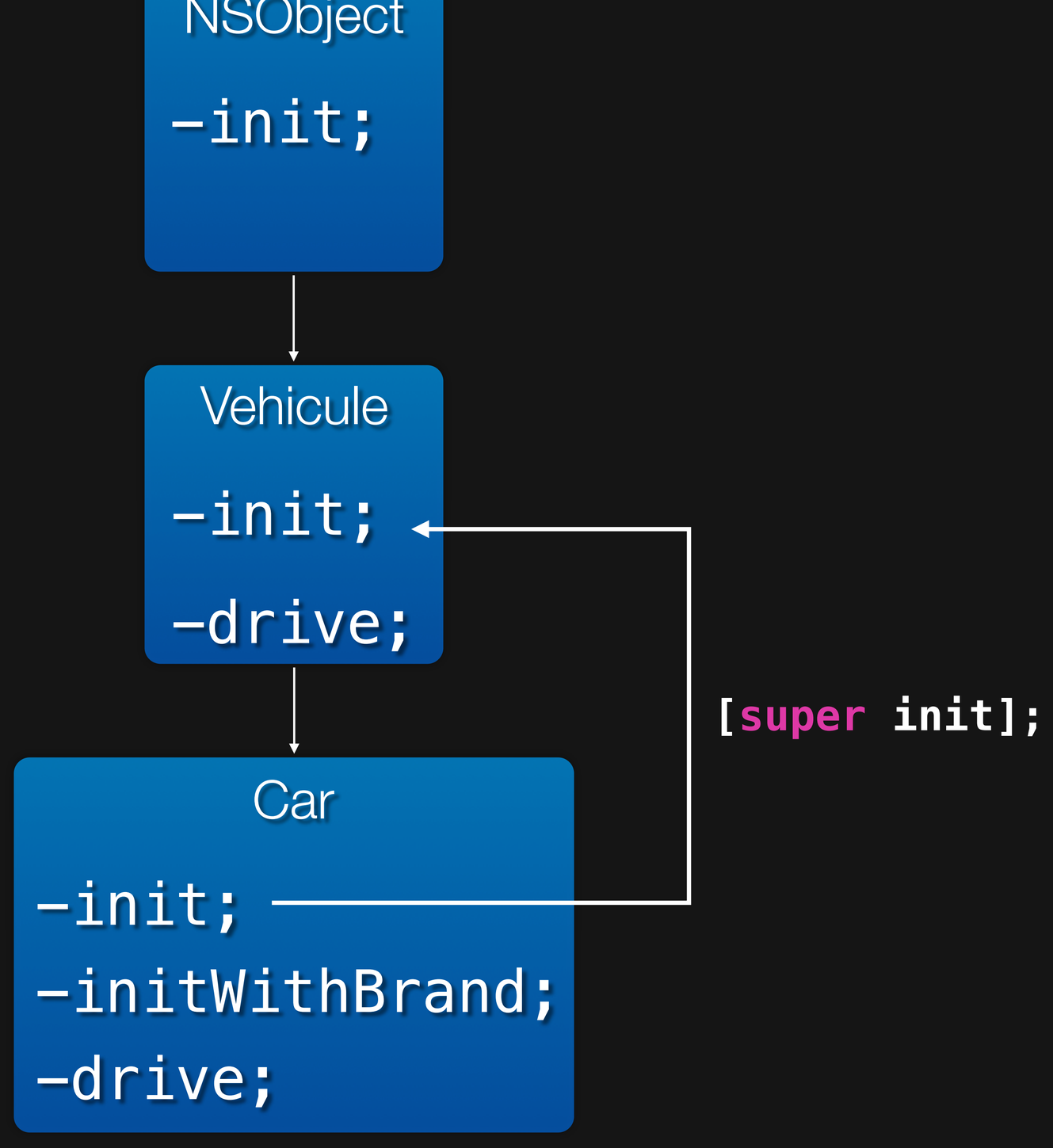
Héritage

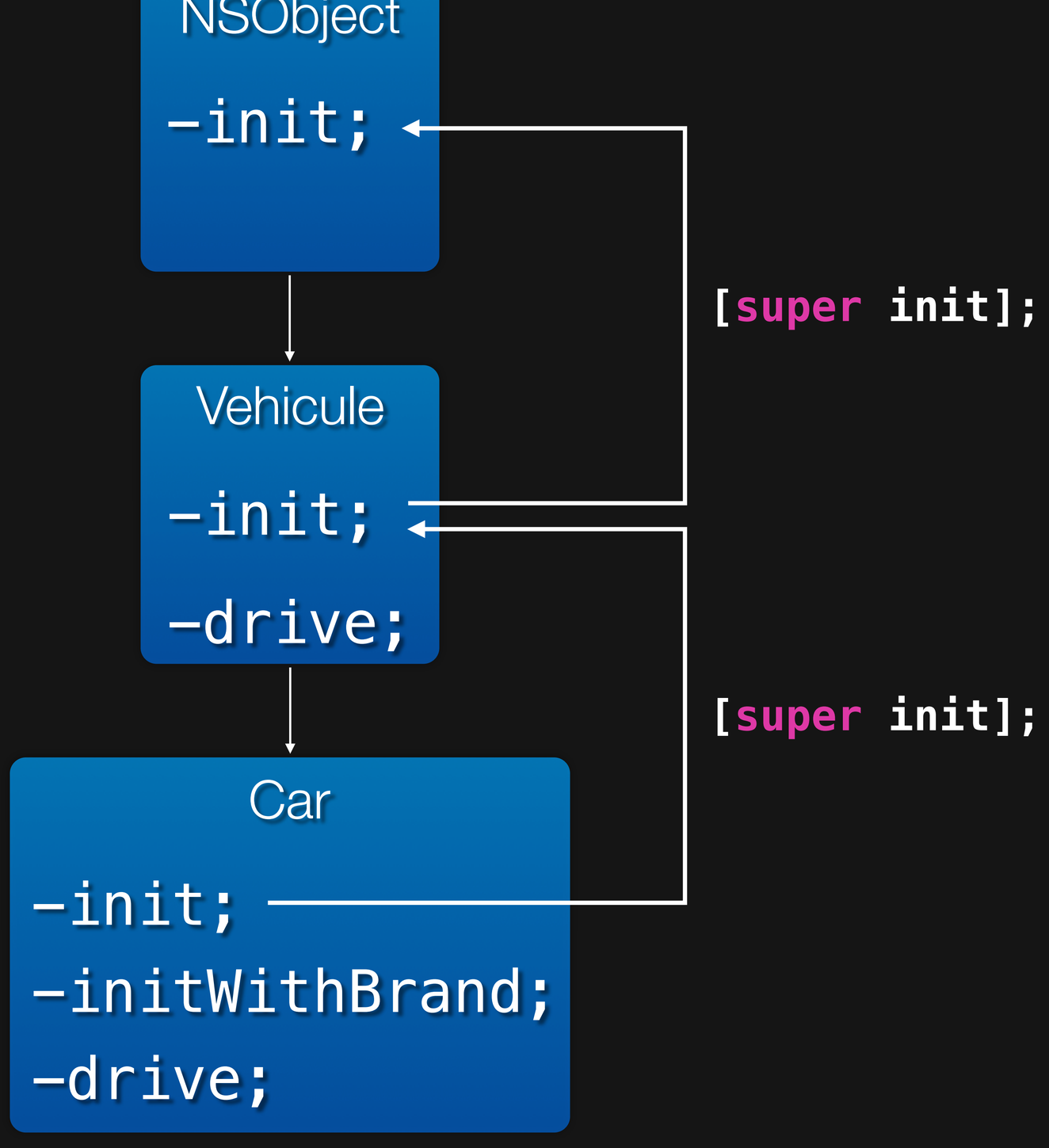
- Objective-C supporte l’héritage simple
- Classe racine principale : NSObject
  - Nos classes doivent hériter de NSObject pour avoir les mécanismes de base d’instanciation

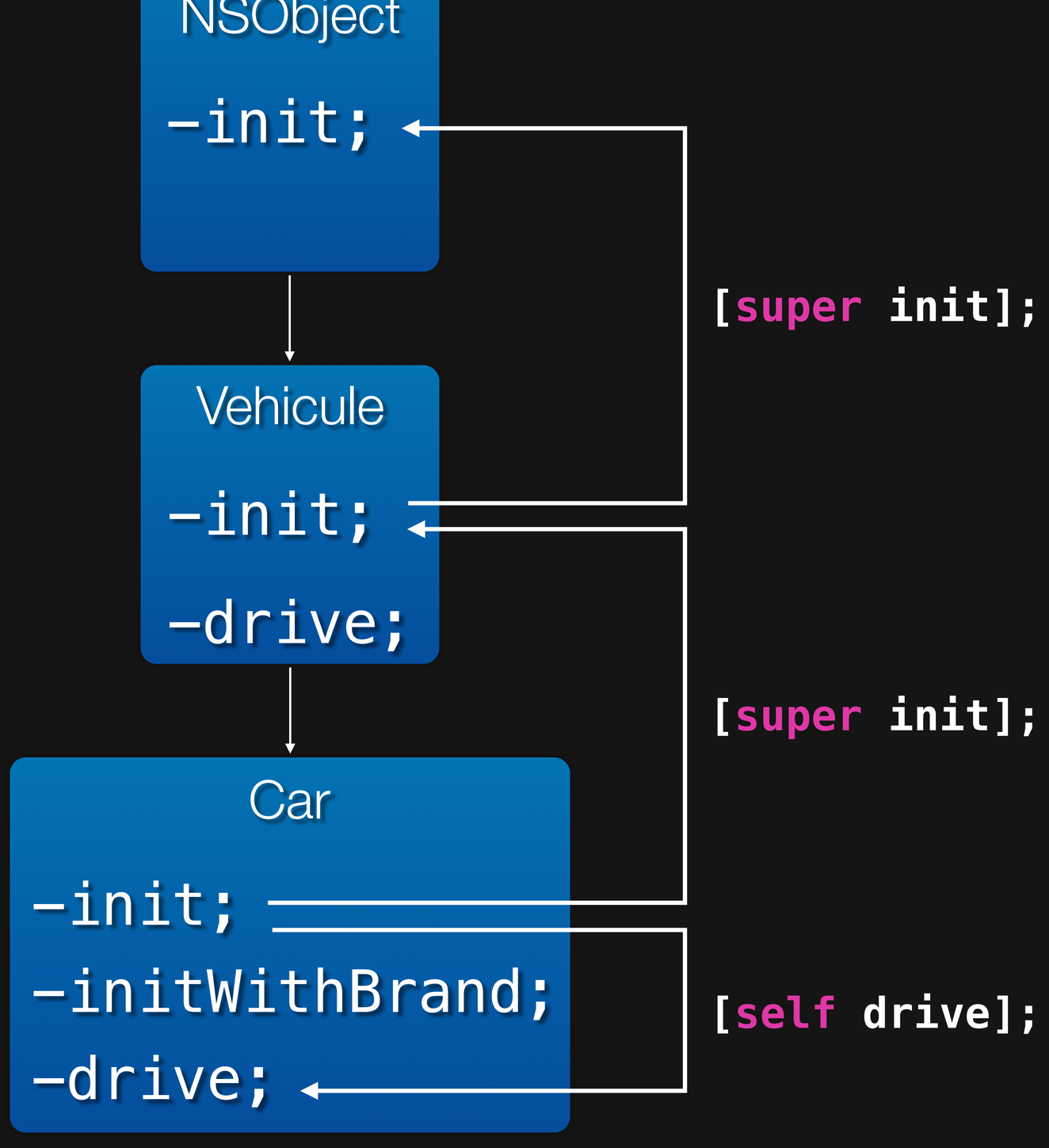
Héritage




NSObject
-init;
Vehicule
-init;
-drive;
Car
-init;
-initWithBrand;
-drive;


NSObject
-init;
Vehicule
-init;
-drive;
Car
-init;
-initWithBrand;
-drive;
[super init];


NSObject
-init;
Vehicule
-init;
-drive;
Car
-init;
-initWithBrand;
-drive;
[super init];
[super init];


NSObject
-init;
Vehicule
-init;
-drive;
Car
-init;
-initWithBrand;
-drive;
[super init];
[super init];
[self drive];

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)

- 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’éxécution (introspection)

Typage dynamique

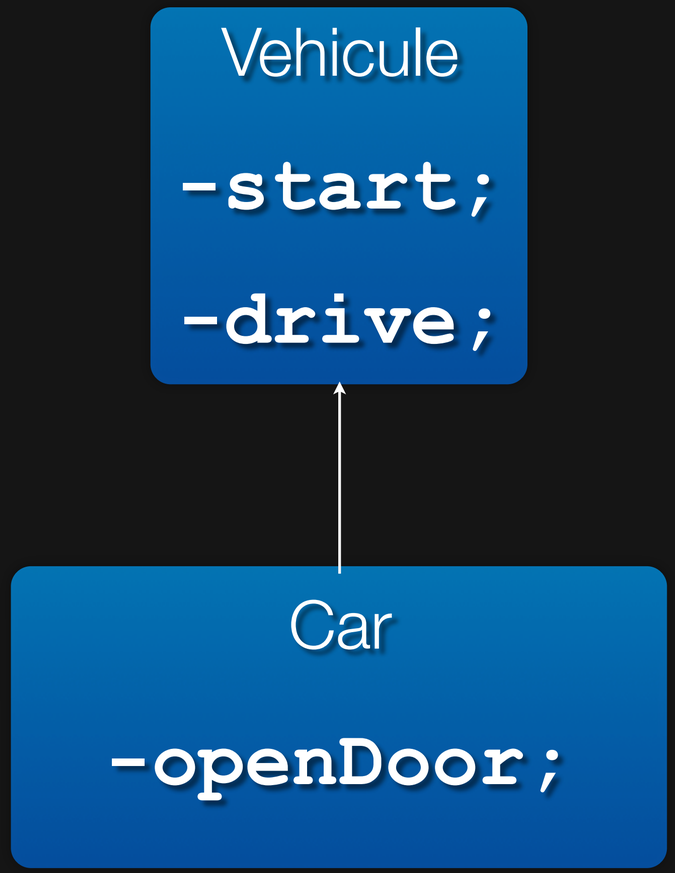
```
NSString *aString = @"This is a string"; // Typage statique
id obj = aString;                        // Typage dynamique
NSArray *anArray = obj;                  // /!\ Attention
```

Typage dynamique

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

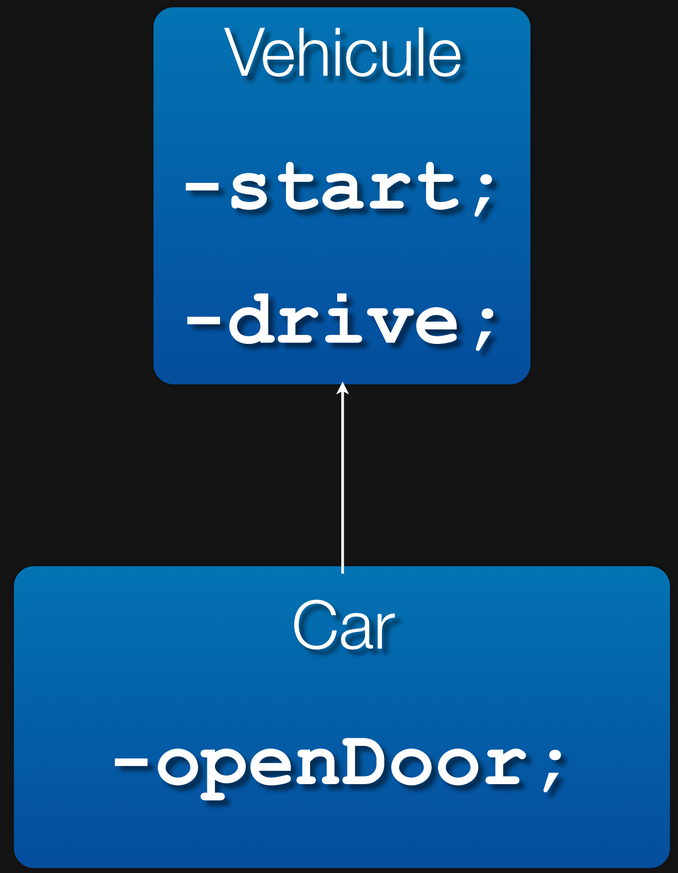
- Préciser le type aide à éviter les erreurs
- Le compilateur peut non prévenir de l’envoi de message “non conforme”

Typage dynamique



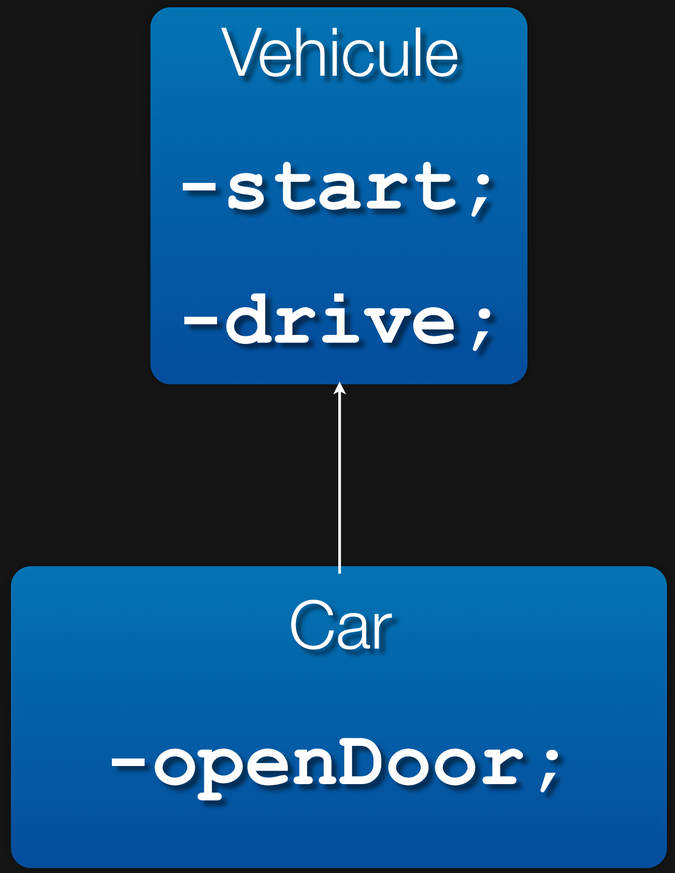
Typage dynamique

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



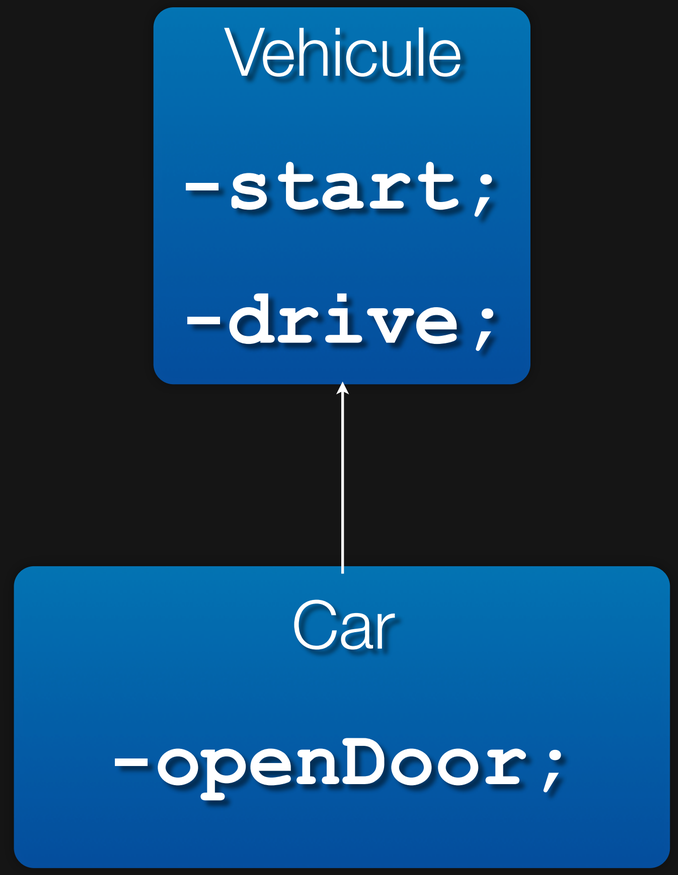
Typage dynamique

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



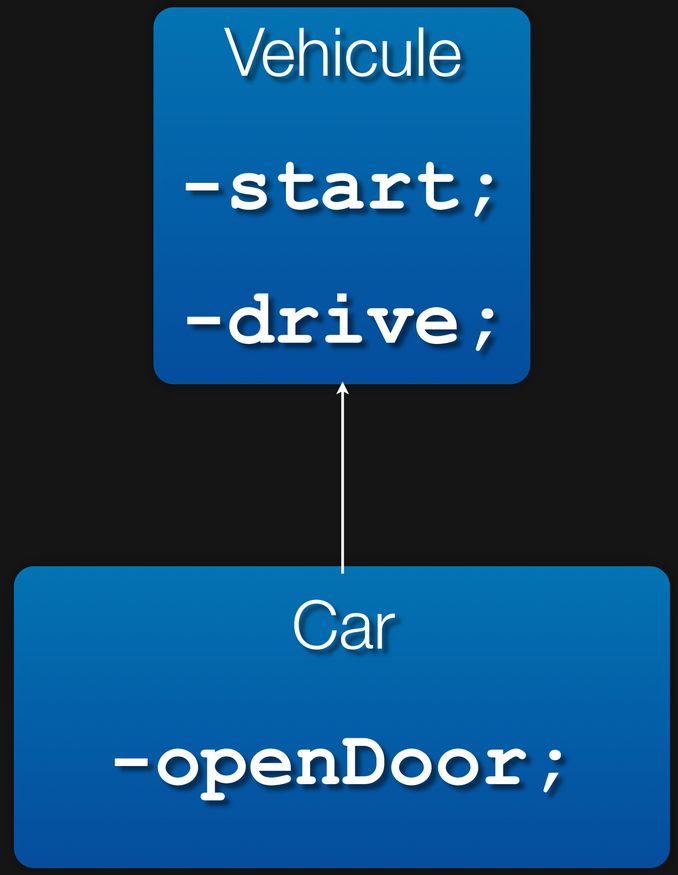
Typage dynamique

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



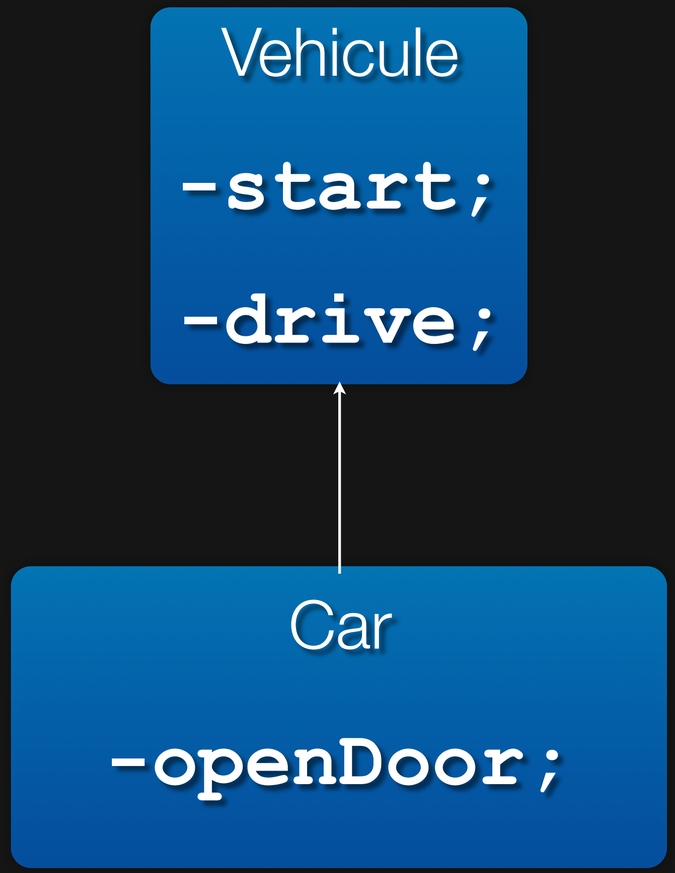
Typage dynamique

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



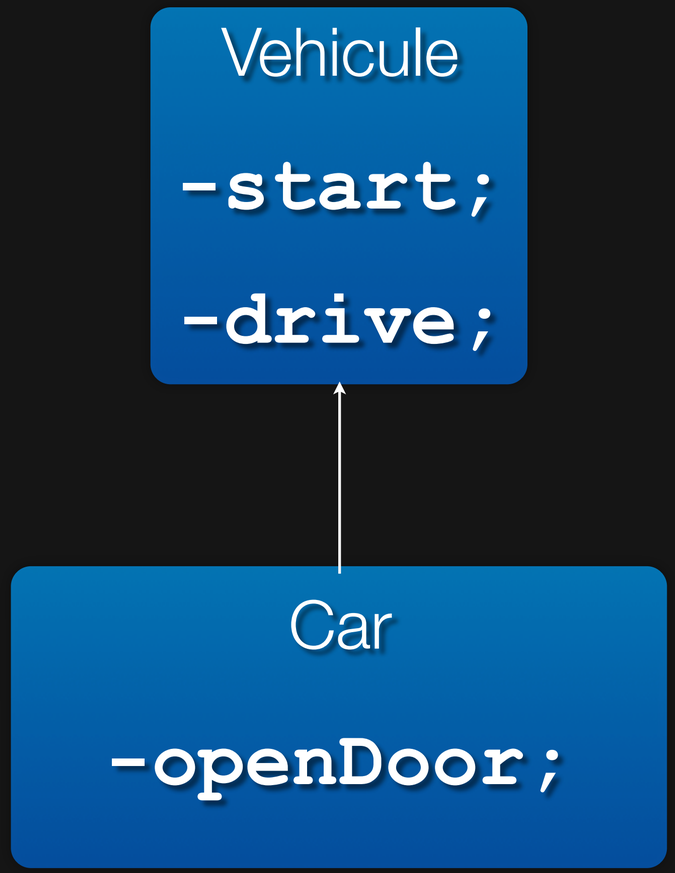
Typage dynamique

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

Vehicule *aVehicule = aCar;
```



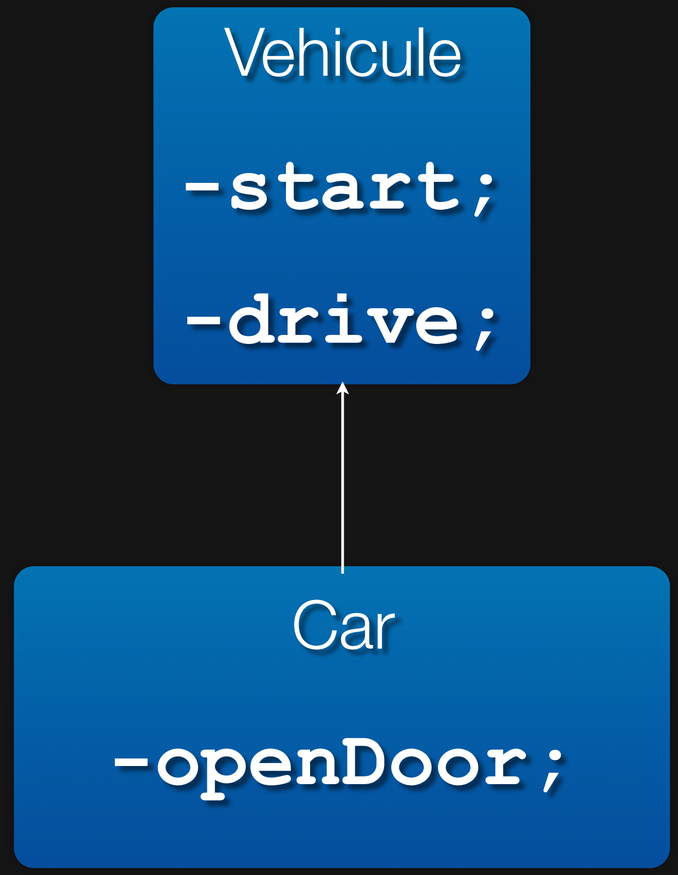
Typage dynamique

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

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



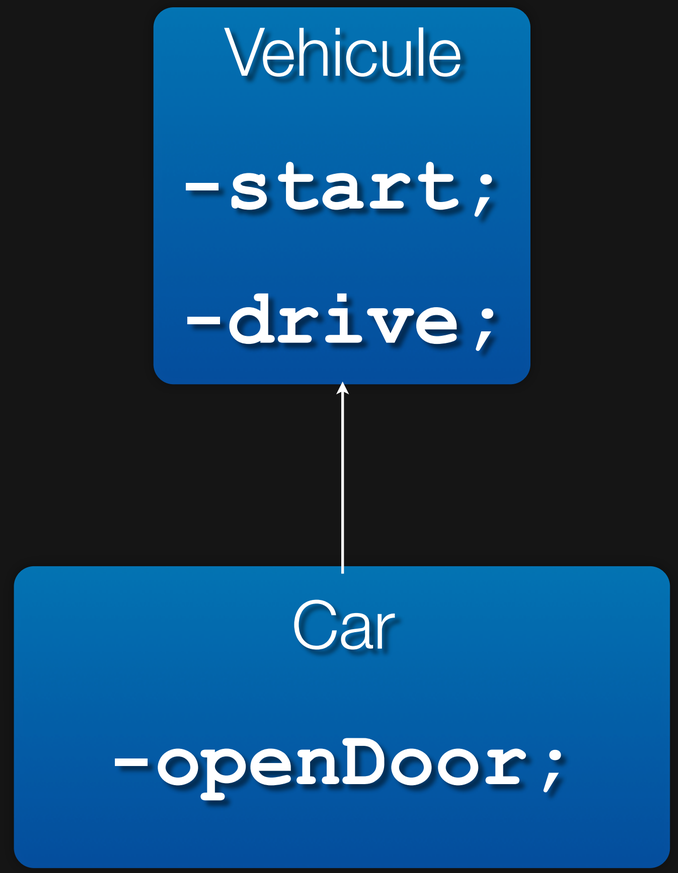
Typage dynamique

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

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



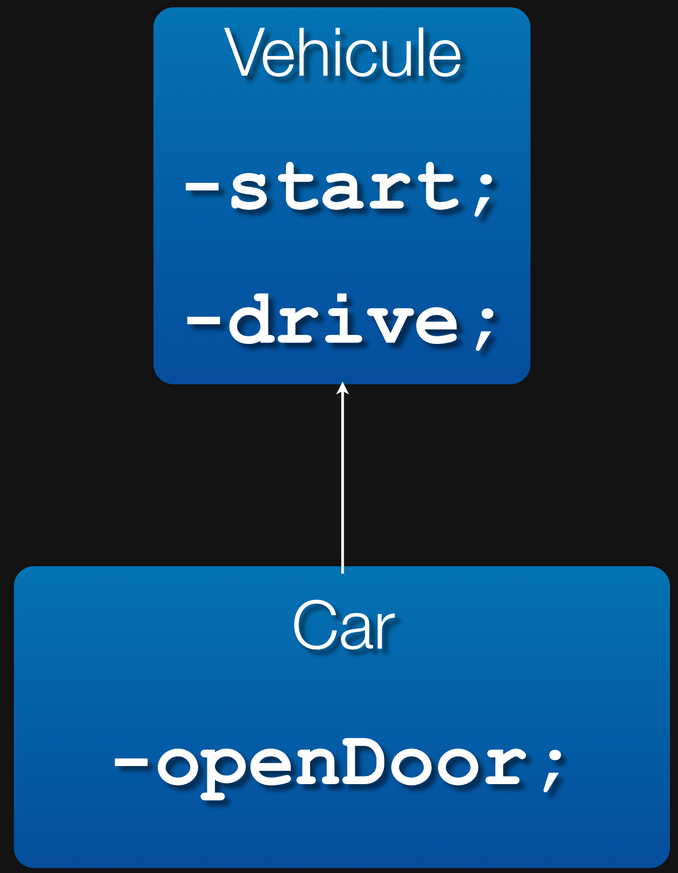
Typage dynamique

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

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



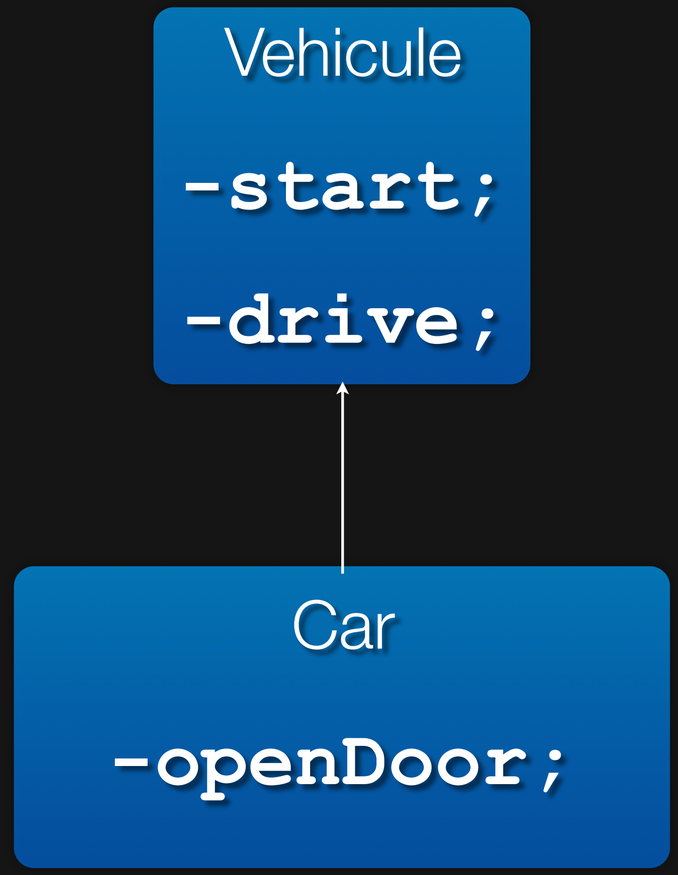
Typage dynamique

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

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



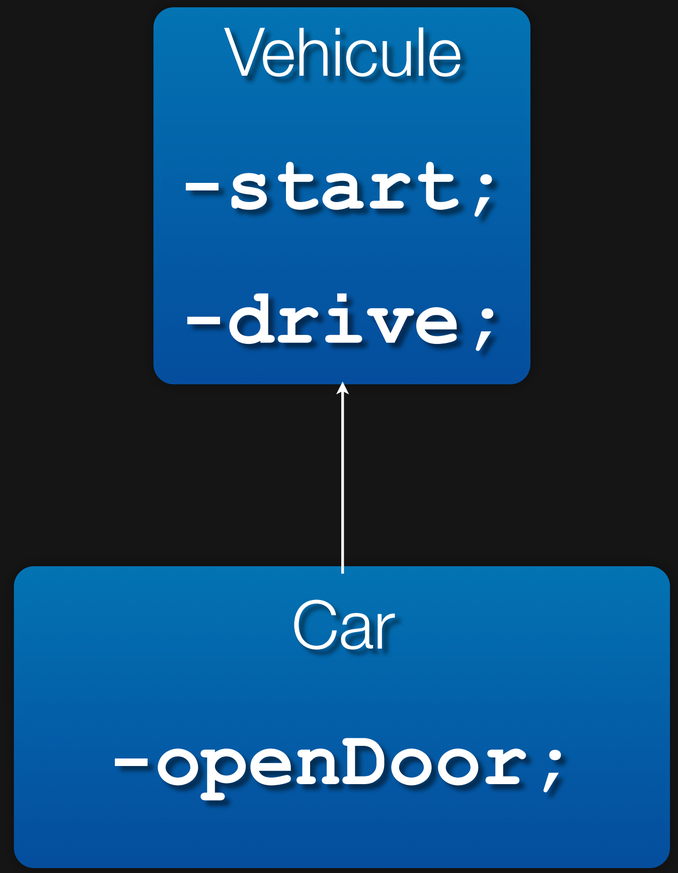
Typage dynamique

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

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



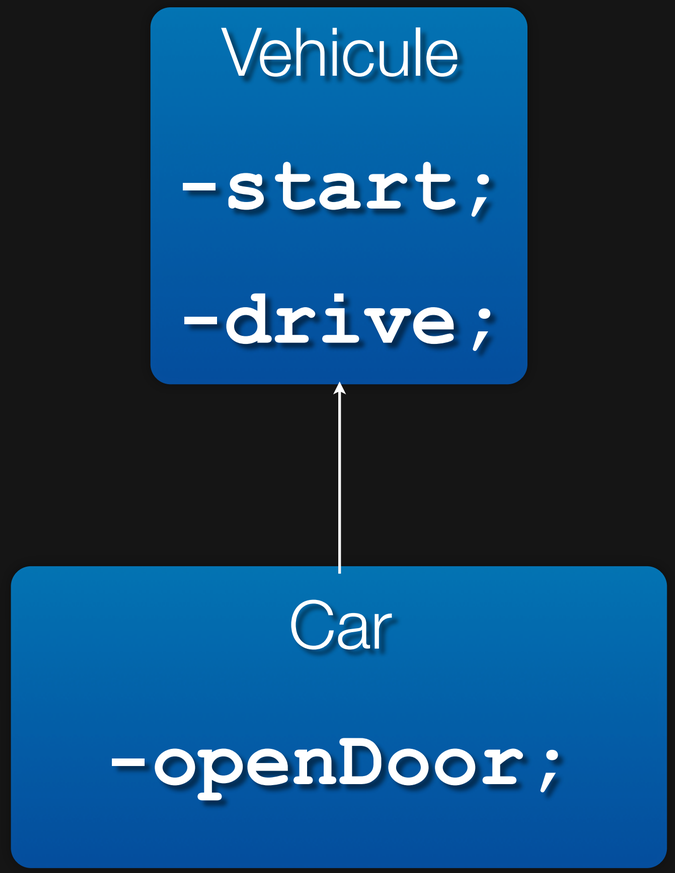
Typage dynamique

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

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



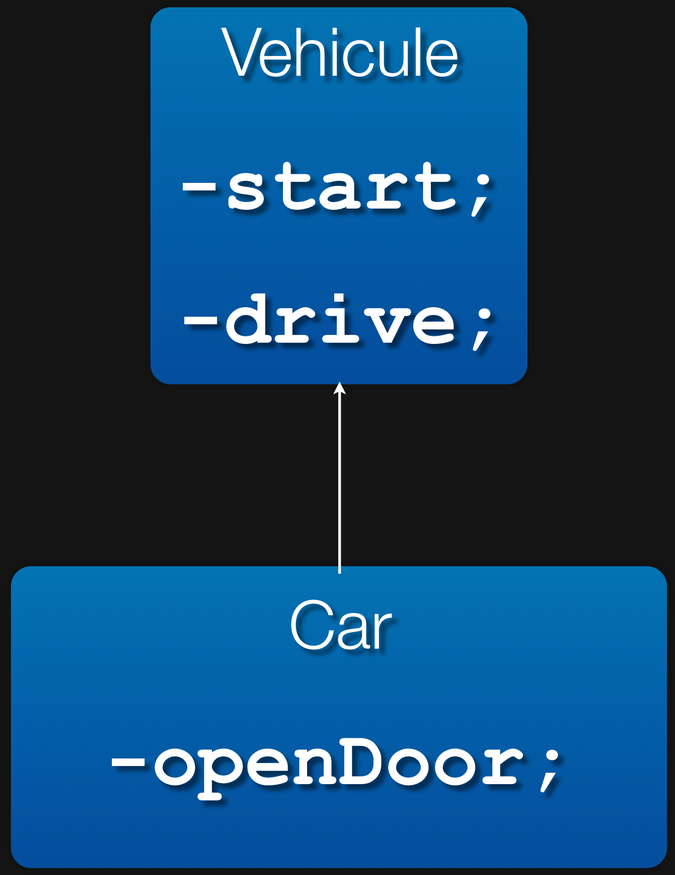
Typage dynamique

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

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



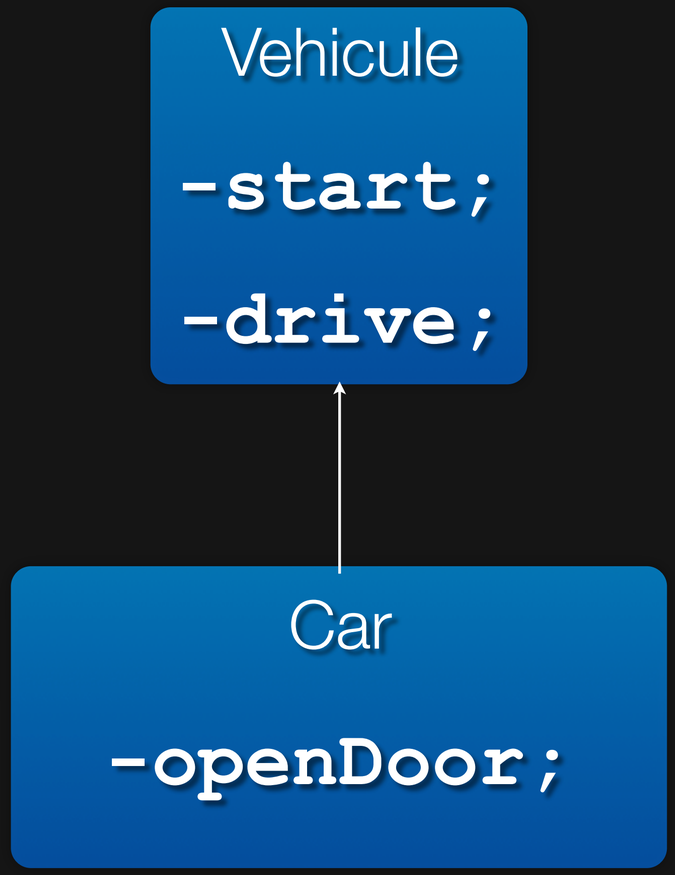
Typage dynamique

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

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



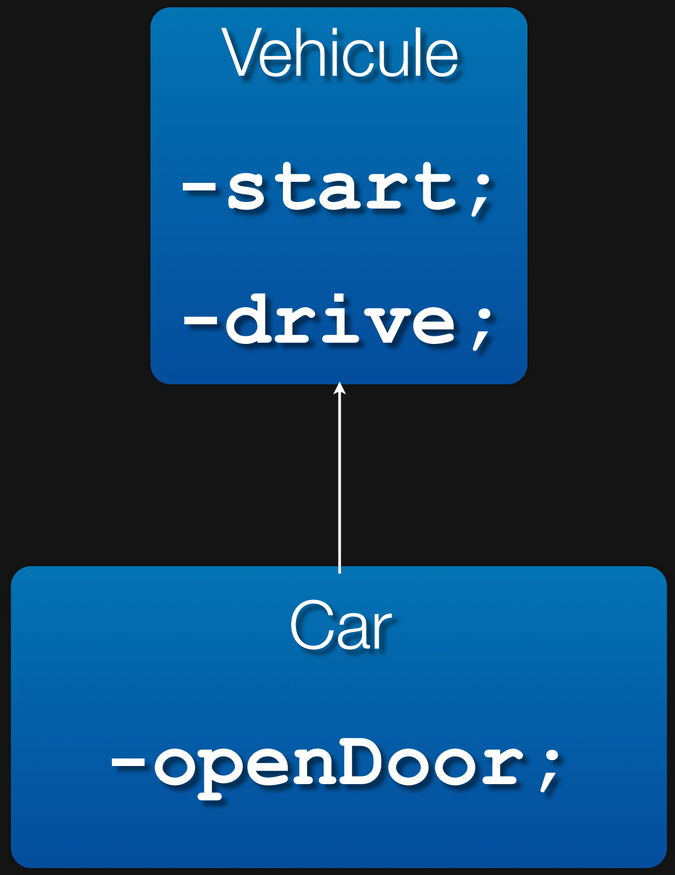
Typage dynamique

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

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



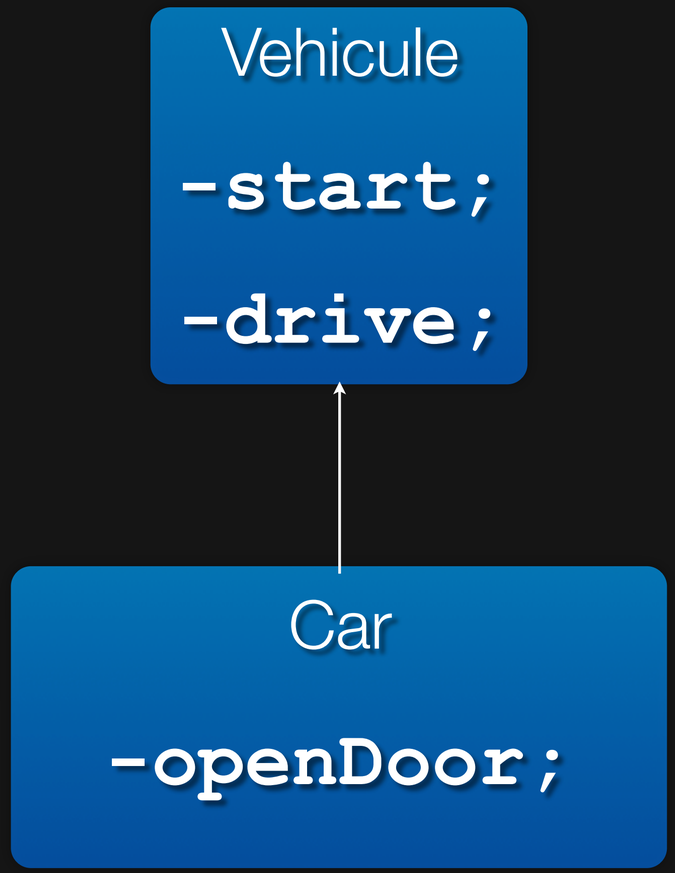
Typage dynamique

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

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



Typage dynamique

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

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

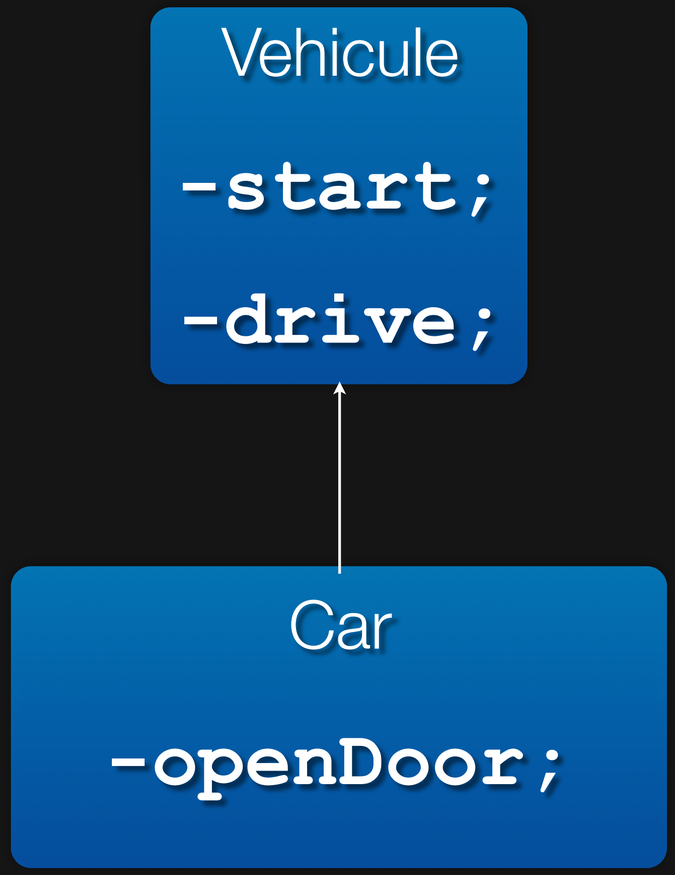


Typage dynamique

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

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

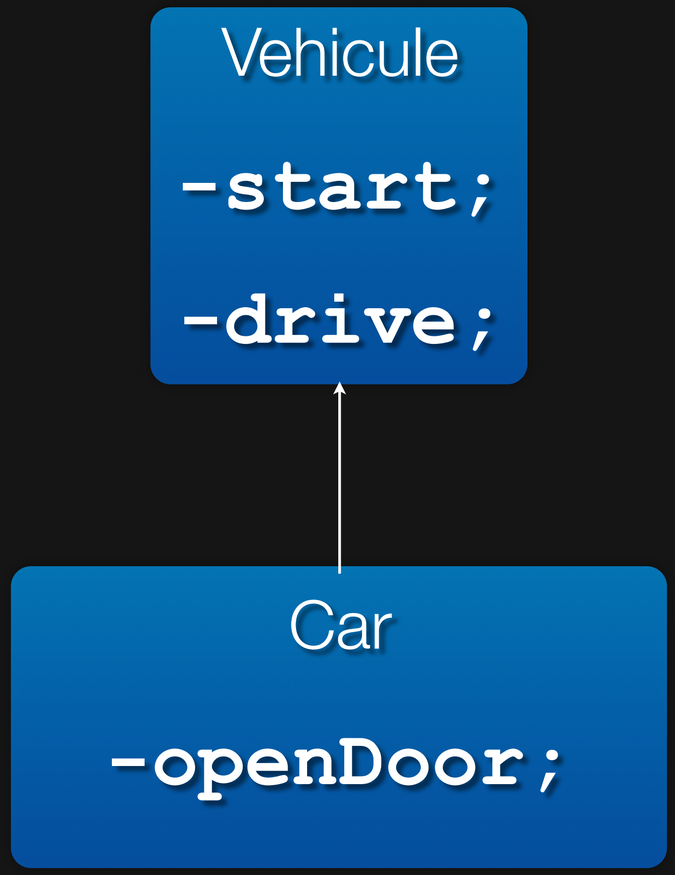
Dog

-bark;



Typage dynamique

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

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



Introspection

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’éxécuter 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];
}
```

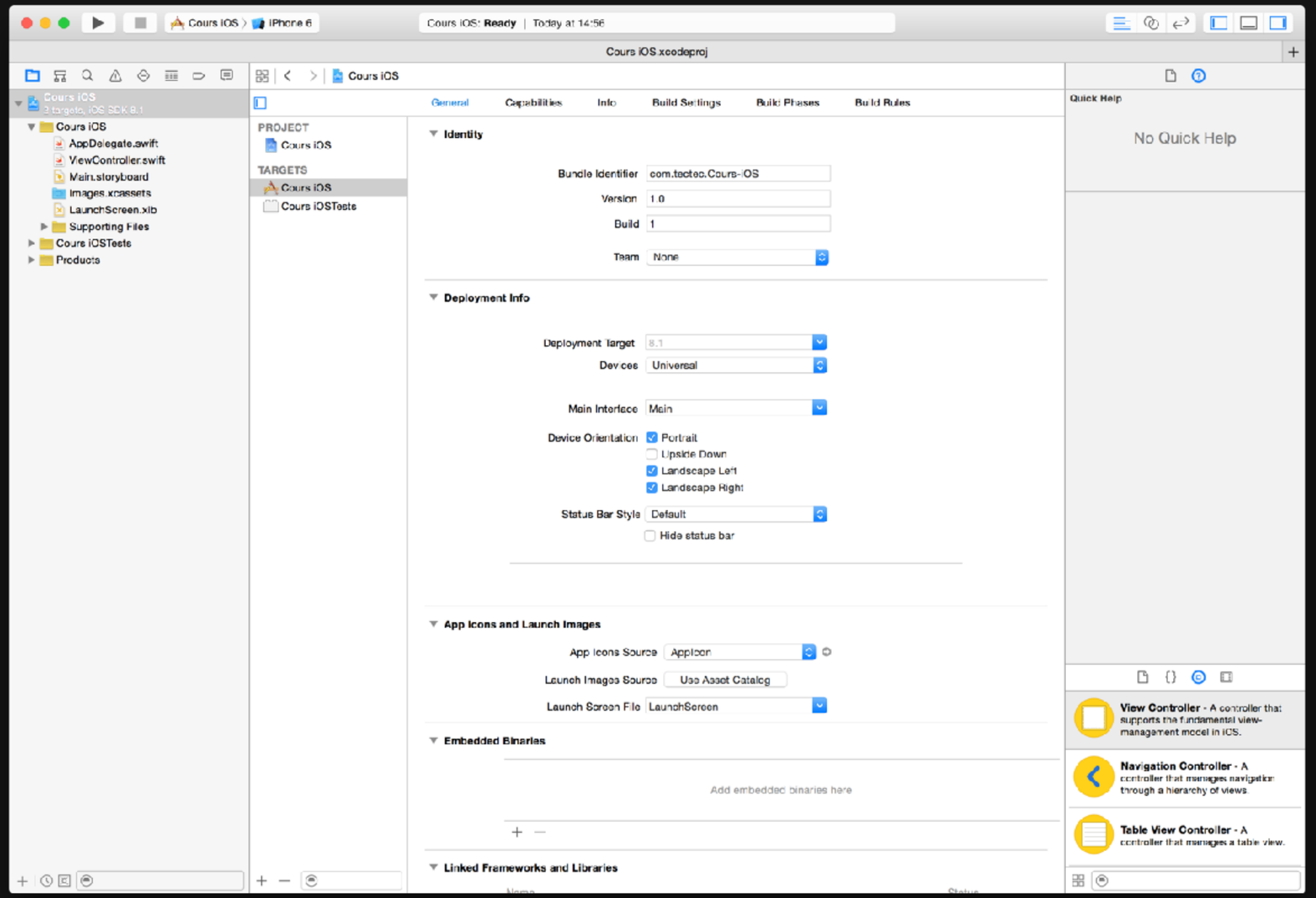
Xcode 7

Xcode 7

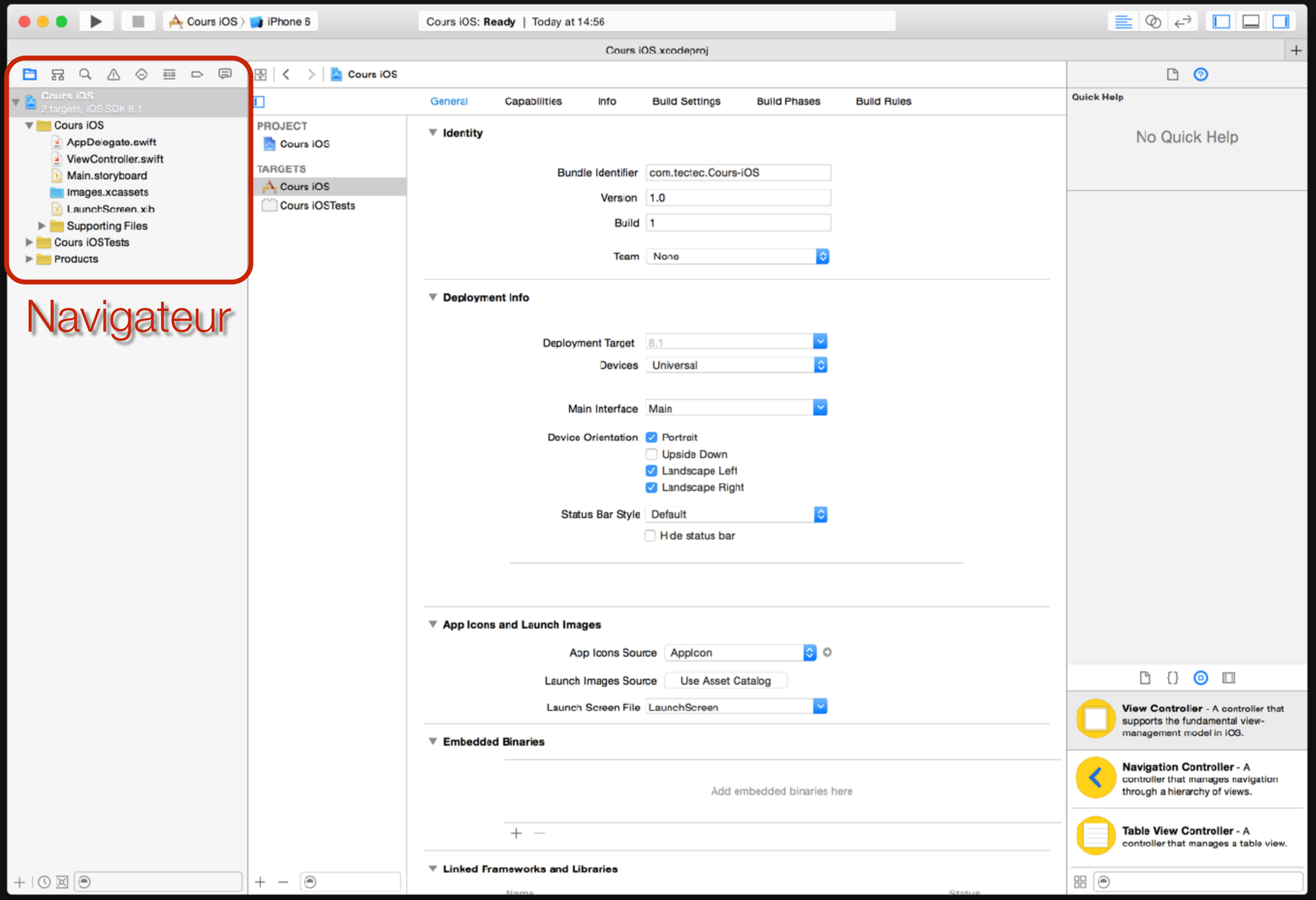
- Xcode 7 à rajouté des fonctionnalités à Objective-C
  - Principalement pour améliorer la compatibilité avec Swift
  - Renforce également la robustesse du code Objective-C
- Annotations de nullabilité
- Génériques légers

Tour de Xcode

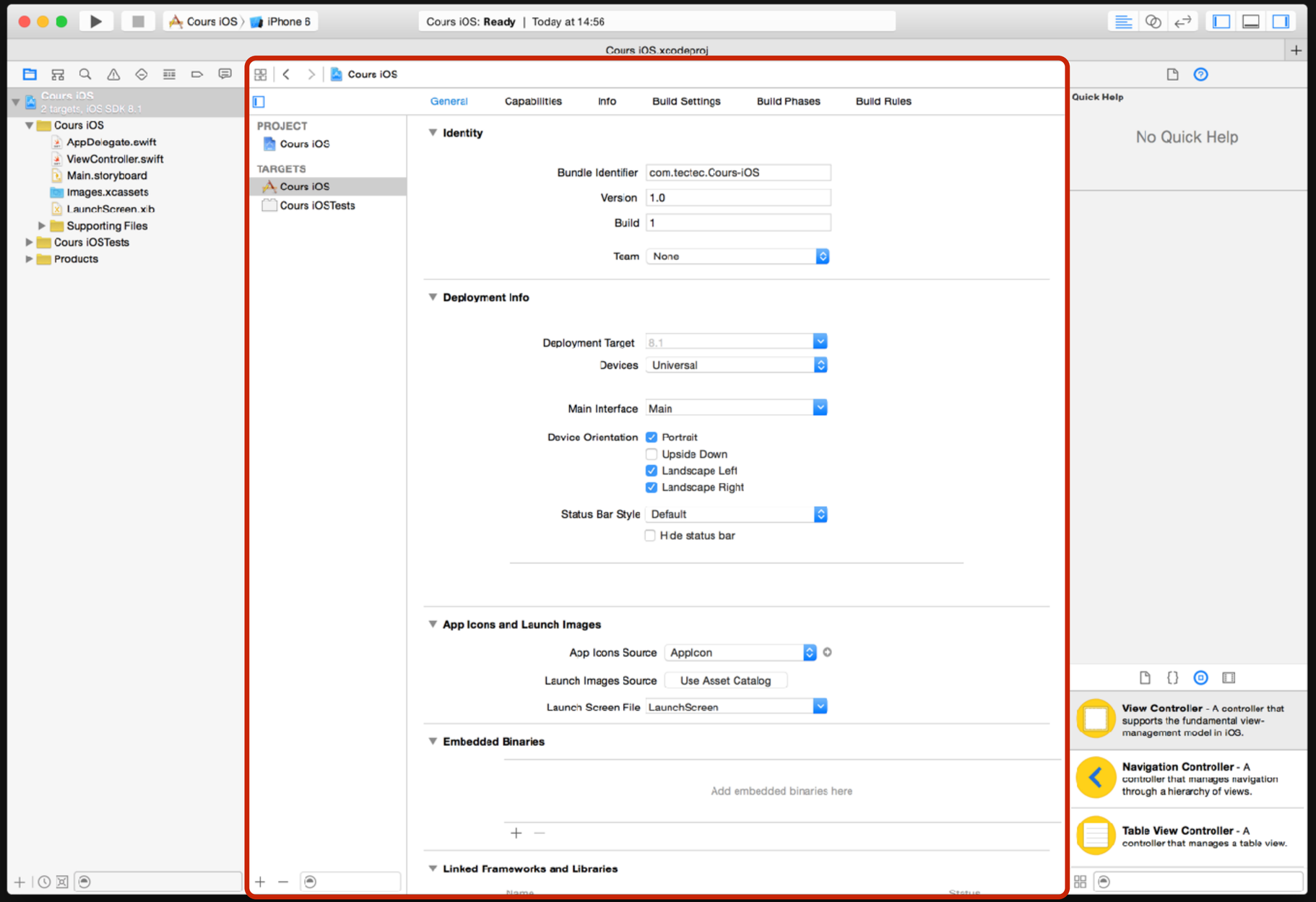
Architecture d'un projet Xcode



Tour de Xcode

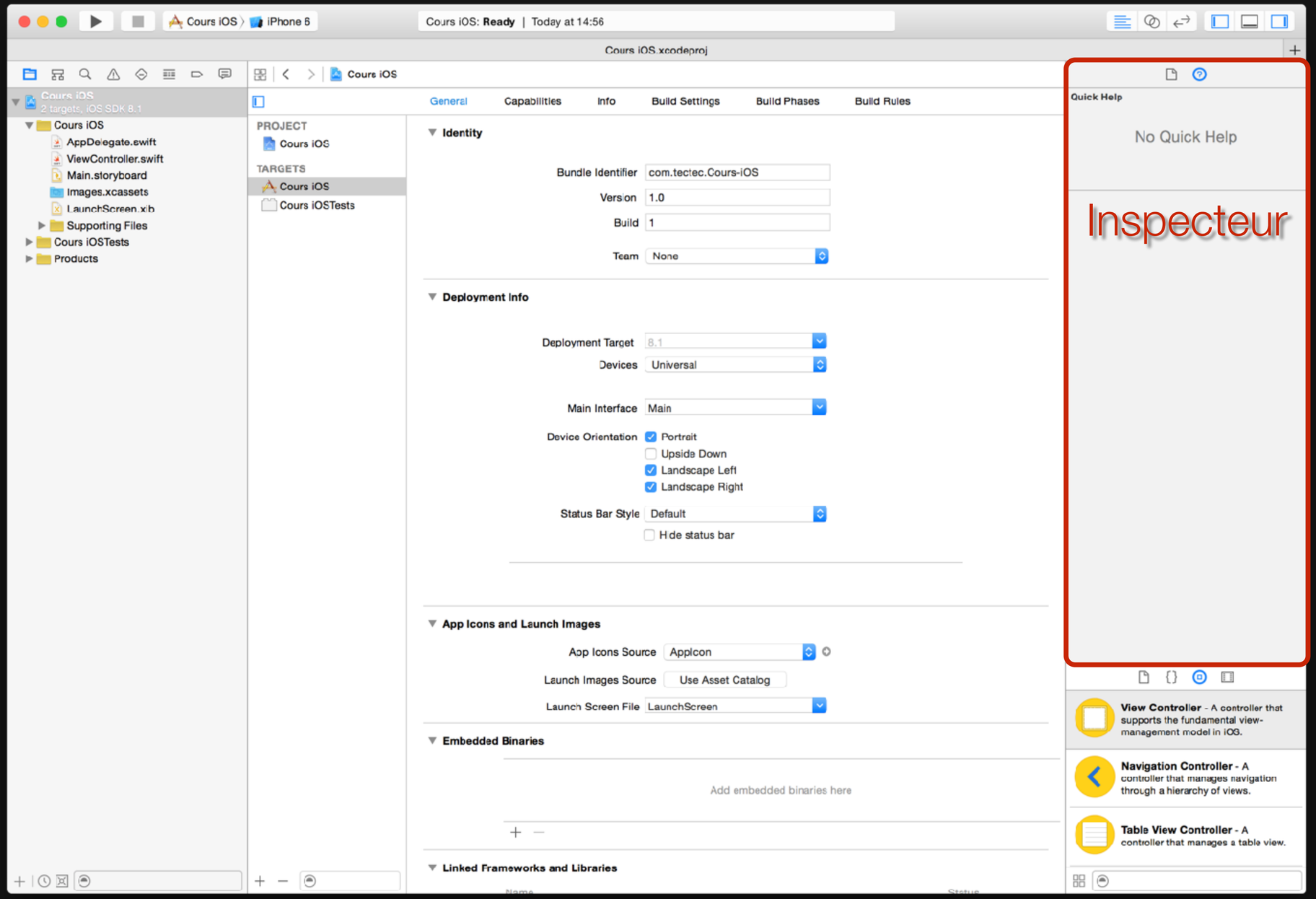


Tour de Xcode

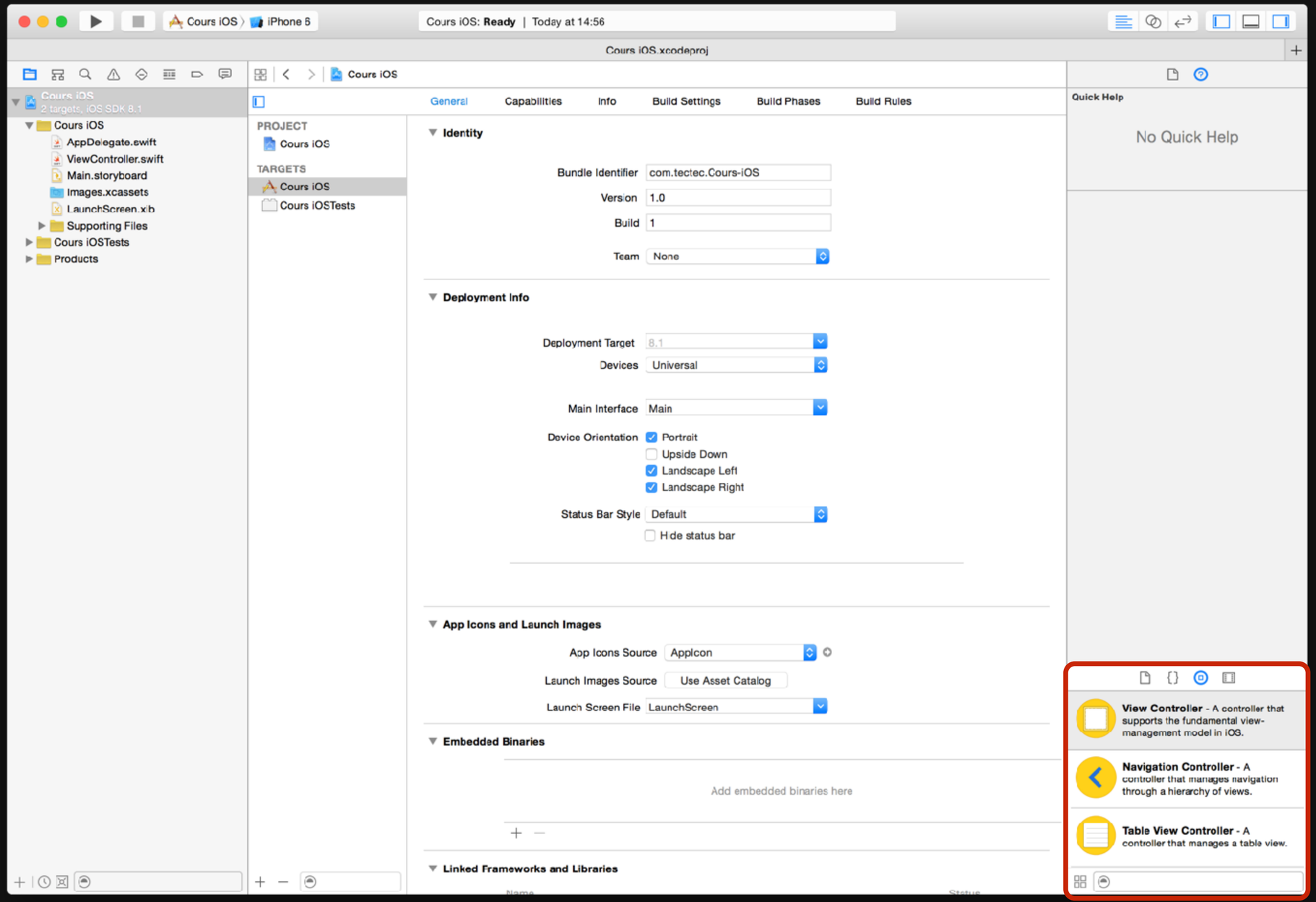


Vue centrale

Tour de Xcode

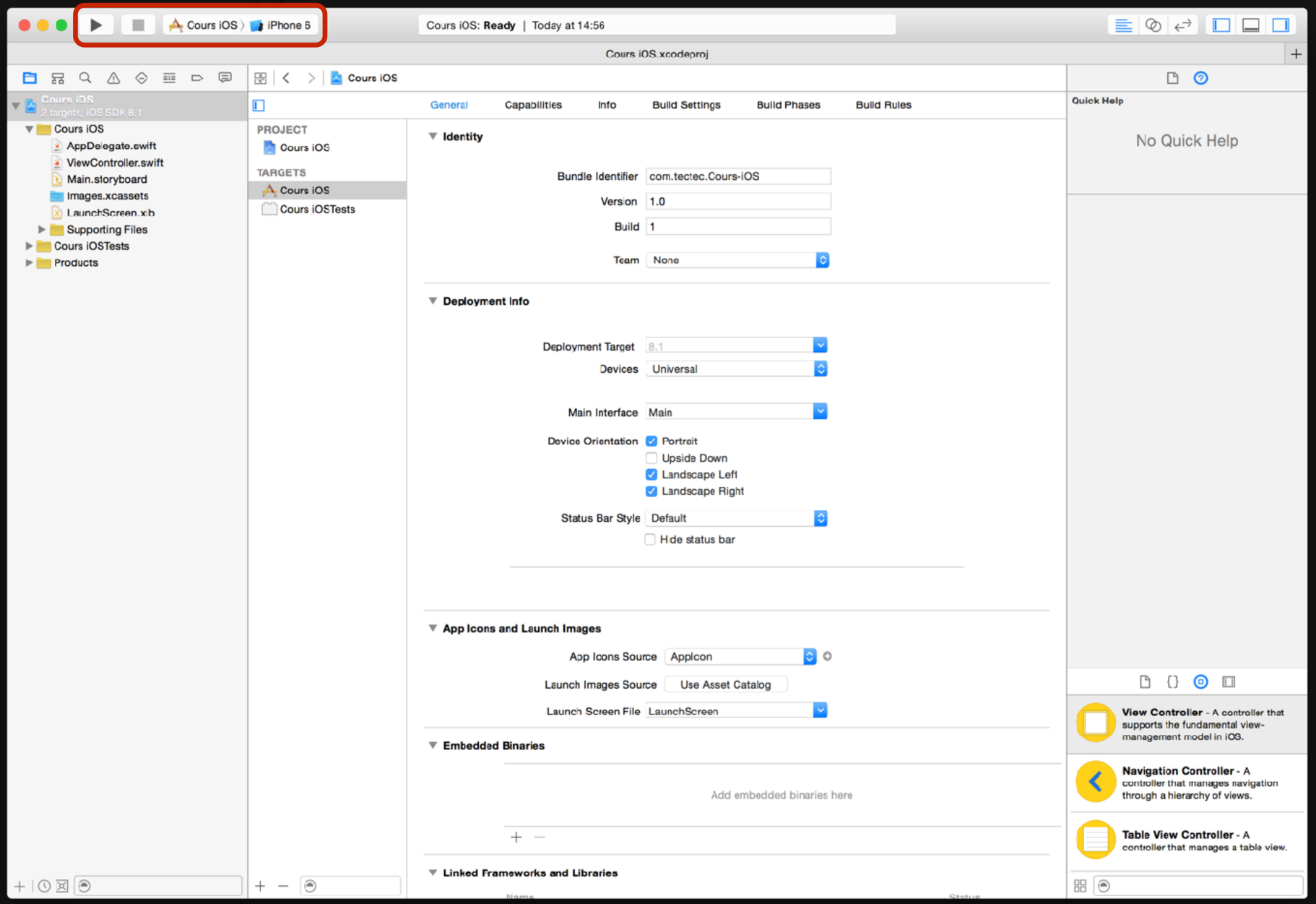


Tour de Xcode

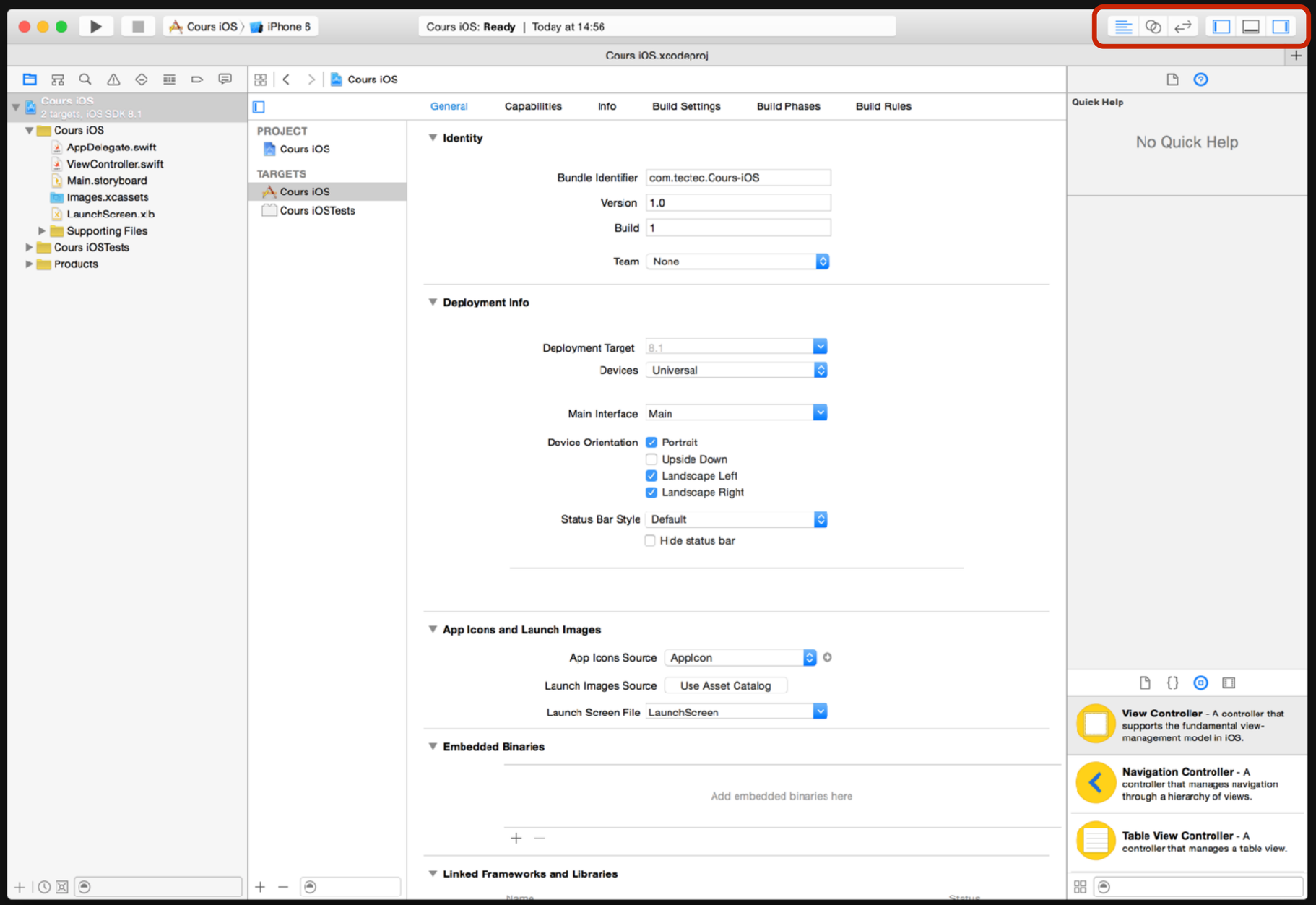


Bibliothèque

Tour de Xcode Contrôle de l'application



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