Objective-C

Sylvain FAY-CHATELARD - IR3

Décembre 2012

1

Sommaire

- Historique
- Les concepts
- La gestion de la mémoire
  - Concepts
  - ARC
  - CoreFoundation
- Conclusion

2

Historique

- 1983
- C, SmallTalk
- Version 2.0 finalisée en 2007
- Principalement utilisé sur les plateformes OSX (Cocoa) et iOS
- A influencé le java en 1995

3

Les concepts

- C’est une surcouche au C
- .h/.m (déclaration, implémentation)
- Multi-plateforme
- Typage dynamique (Type vérifié à l’exécution)
- Typage faible (int a; int b; short c=a+b)

4

Les concepts

- Qu'est-ce qu'on a en plus? (Objective-C 2.0)

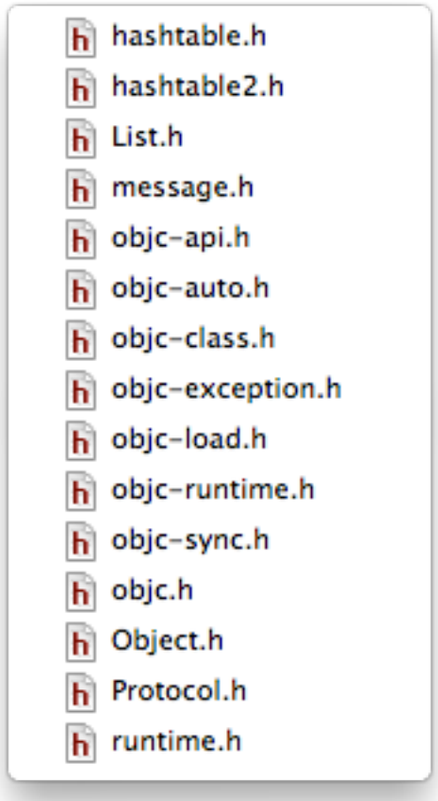
```
typedef struct objc_class *Class;
typedef struct objc_object {
    Class isa;
} *id;

...


#define YES             ((BOOL)1)
#define NO              ((BOOL)0)

#ifndef Nil
# if __has_feature(cxx_nullptr)
#   define Nil nullptr
# else
#   define Nil __DARWIN_NULL
# endif
#endif

#ifndef nil
# if __has_feature(cxx_nullptr)
#   define nil nullptr
# else
#   define nil __DARWIN_NULL
# endif
```



/usr/include/objc/

5

Les concepts

- Les fonctions (message)

```
[receiver message];
```

```
- (void)display;
[myRectangle display];
```

```
- (void)setWidth:(double)width;
[myRectangle setWidth:20.0];
```

```
- (void)setOriginX:(double)x y:(double)y;
[myRectangle setOriginX: 30.0 y: 50.0];
```


6

Les concepts

- Les classes

.h

```objc
#import <UIKit/UIKit.h>

@interface Object : NSObject <ObjectDelegate>

@property (strong, nonatomic) NSString *attribut;

-(void)methodPublic;

+(void)methodStatic;

@end
```

.m

```objc
#import "AppDelegate.h"

@implementation AppDelegate

-(void)methodPublic
{
    NSLog(@"public %@", _attribut);
}

+(void)methodStatic
{
    NSLog(@"static");
}

@end
```


7

Les concepts

- Les protocoles (Implémentation)

DetailViewController.h

```
@protocol DetailDelegate <NSObject>

-(void)methodDelegate:(NSString*)plop;

@end

@interface DetailViewController : UIViewController

@property (strong, nonatomic) id<DetailDelegate> delegate;
```

DetailViewController.m

```
- (void)callDelegate
{
    if (_delegate) {
        NSLog(@"Delegate call from %p", self);
        [_delegate methodDelegate:@"Youpi"];
    }
}
```


8

Les concepts

- Les protocoles (Utilisation)

On signifie qu'on implémente la délégation

```
@interface MasterViewController : UITableViewController <DetailDelegate>
```

Lorsqu'on crée l'objet, on s'inscrit à la délégation

```
DetailViewController *d = [[DetailViewController alloc] init];
[d setDelegate:self];  // ou d.delegate = self;
```

On implémente la méthode déléguée

```
- (void)methodDelegate:(NSString *)plop
{
    NSLog(@"Call by %p with value %@", self, plop);
}
```


9

Les concepts

- Le sub-classing

Permet l'implémentation de nouvelles méthodes à un objet auquel on n'a pas accès

NSArray+OtherMethod.h

NSArray+OtherMethod.h

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

@interface NSArray (OtherMethod)

- (void)methodAdded:(BOOL)isAdded;

@end
```

NSArray+OtherMethod.m

NSArray+OtherMethod.m

```objc
#import "NSArray+OtherMethod.h"

@implementation NSArray (OtherMethod)

- (void)methodAdded:(BOOL)isAdded
{
    NSLog(@"Method is added ? %d",
isAdded);
}

@end
```


10

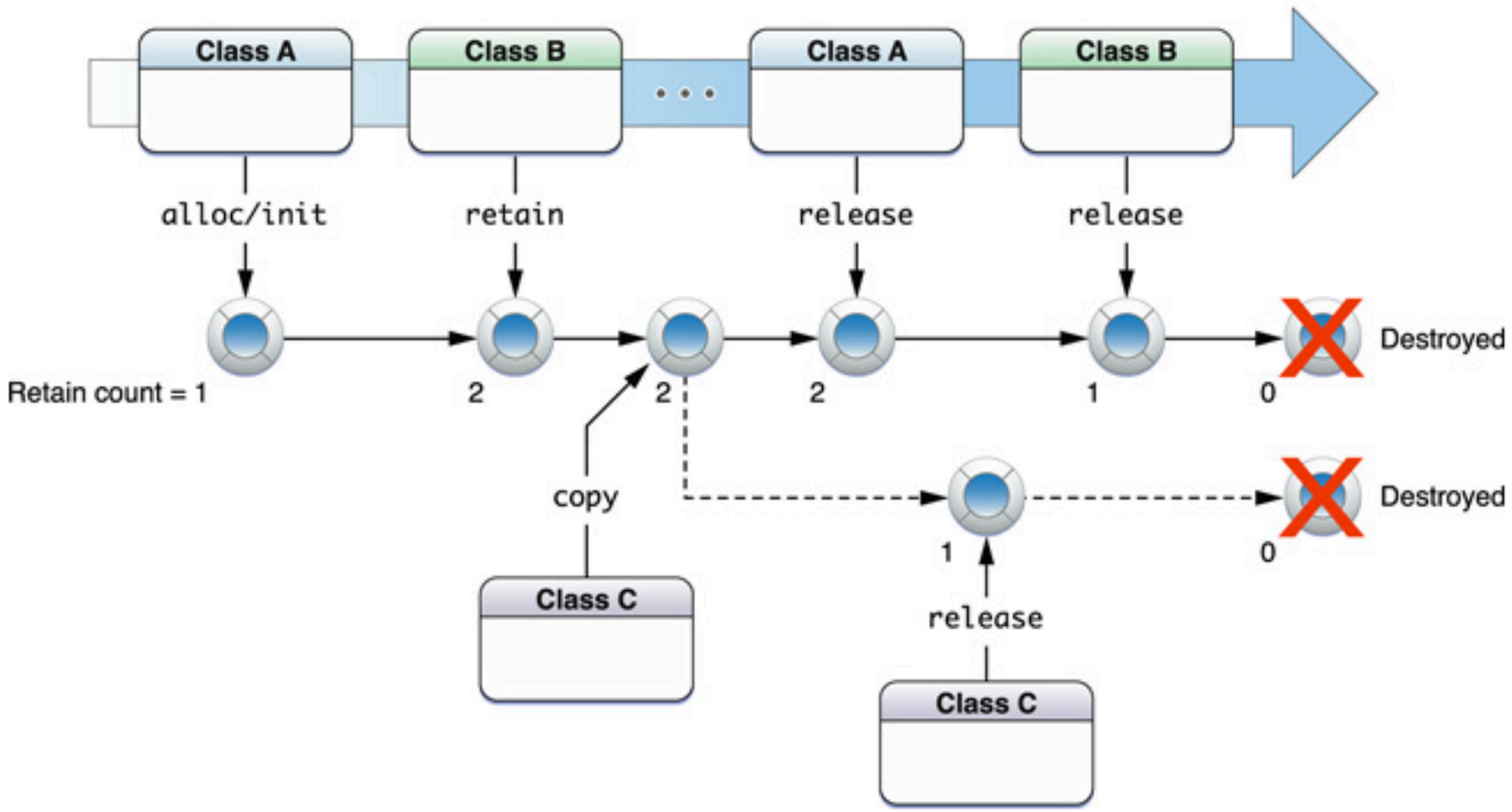
La gestion de la mémoire

- Le concept de compteur et les méthodes associées
- L'Automatic Reference Counting (ARC)
- La mémoire avec CoreFoundation

11

La gestion de la mémoire

Compteur de référence
Méthode retain et release



12

La gestion de la mémoire

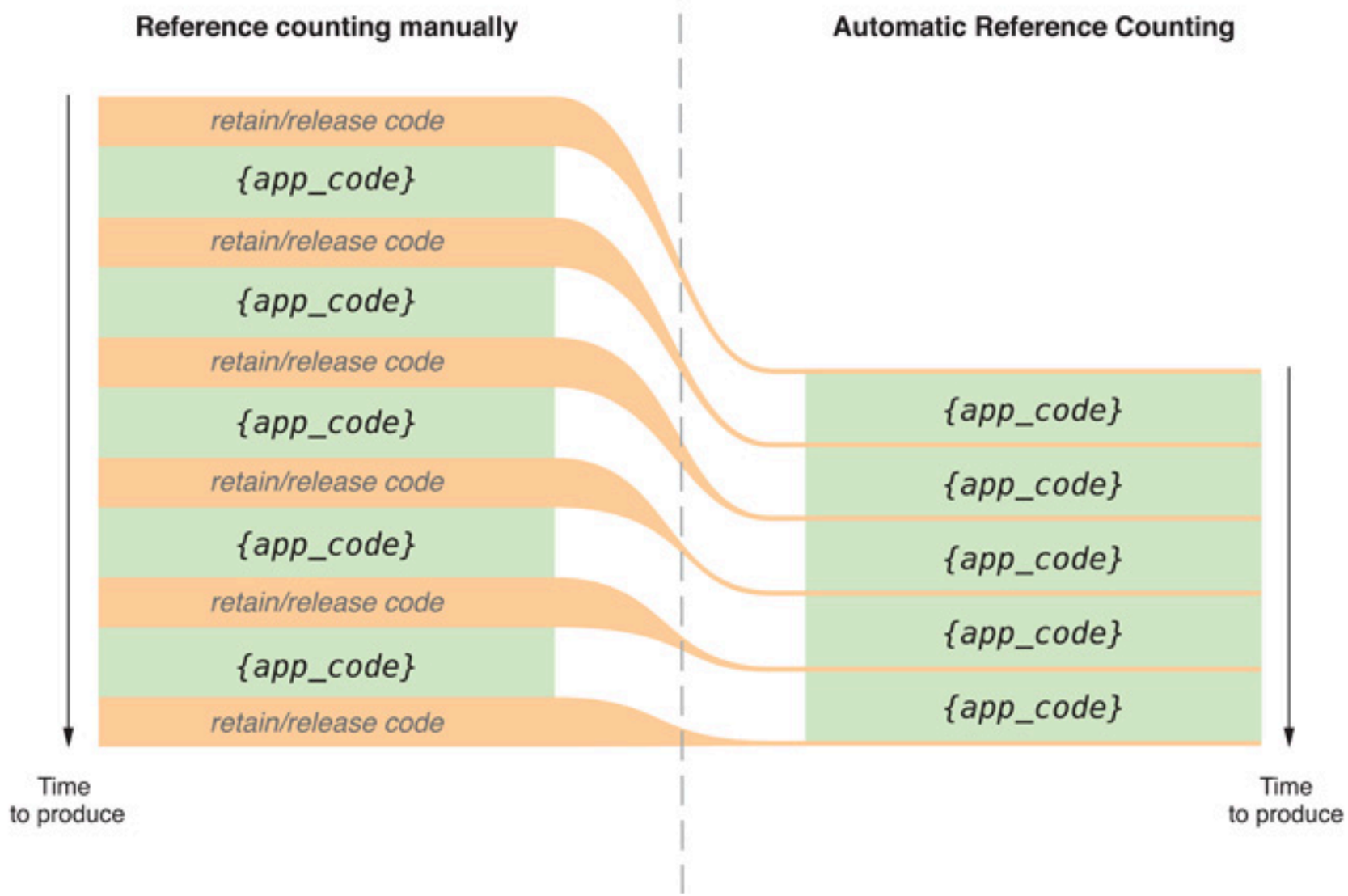
Automatic Reference Counting

- On n’a plus rien à faire

13

La gestion de la mémoire

Automatic Reference Counting



14

La gestion de la mémoire

Automatic Reference Counting

- Quelques points importants si on souhaite affiner :
    - Deux types de pointeurs (weak/strong)

15

La gestion de la mémoire

Automatic Reference Counting

- Weak : Pointeur léger, désalloué au plus vite

```
NSString * __weak str = @"Youpi";
```


16

La gestion de la mémoire

Automatic Reference Counting

- Strong : Pointeur fort, persistant (par défaut)

```
NSString * __strong str = @"Youpi";
```


17

La gestion de la mémoire

Avec Core Foundation

On garde la gestion de la mémoire à la main mais :

```
CFStringRef str = CFSTR("Hello, World!\n");
CFRetain(str);
CFRelease(str);
```


18

Bibliographie

- http://clang.llvm.org/docs/AutomaticReferenceCounting.html
- Documents disponibles sur le site développeur d'Apple :
  - The Objective-C Programming Language
  - Memory Usage Performance Guideline
  - Advanced Memory Management Programming Guide
  - Memory Management Programming Guide For Core Foundation
  - Transitionning to ARC Release Notes

19