SCIENCES SORBONNE UNIVERSITÉ

Objective-C, the reference counter

Fabrice.Kordon@lip6.fr

LIP6

F. Kordon - Sorbonne Université - CC2018

Principles (1/2)

Each object embed a reference counter

- Defined in NSObject
  - If counter > 0, the object lives
  - if counter ≤ 0, the object is destroyed

Manipulation this counter

- +alloc
  - Creates objects with counter = 1
- -copy
  - Returns an object with counter = 1 (∀ copied object)
- -retain
  - Increments the counter
- -release
  - Decrements the counter

F. Kordon - Sorbonne Université - CC2018

Principles (2/2)

As soon as a counter = 0

- It can be destroyed anytime
  - **iOS decides**
  - **Beware, the simulator is «nicer»**
- Never let this happens if you still need the object

If counter > 0

- It remains in memory

Autorelease

- Will be explained in a dedicated video

F. Kordon - Sorbonne Université - CC2018

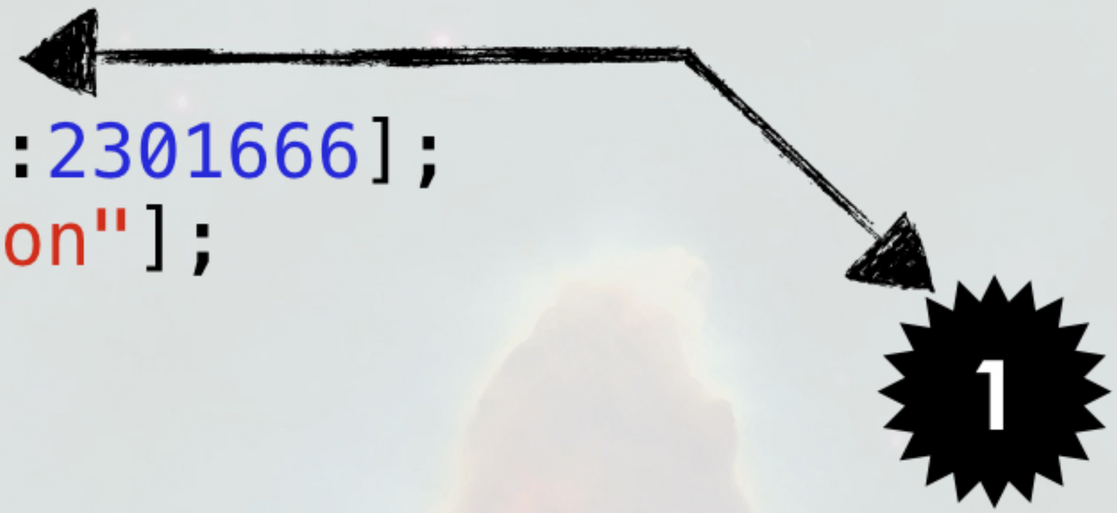
An example


```
Student *aStudent;

aStudent = [[Student alloc]
                initWithNumber:2301666];
[aStudent setName:@"tartempion"];
[aStudent retain];

NSLog(@"Nouvel étudiant : %@",
      [aStudent identity]);

[aStudent release];

[aStudent release];
```

F. Kordon - Sorbonne Université - CC2018

An example

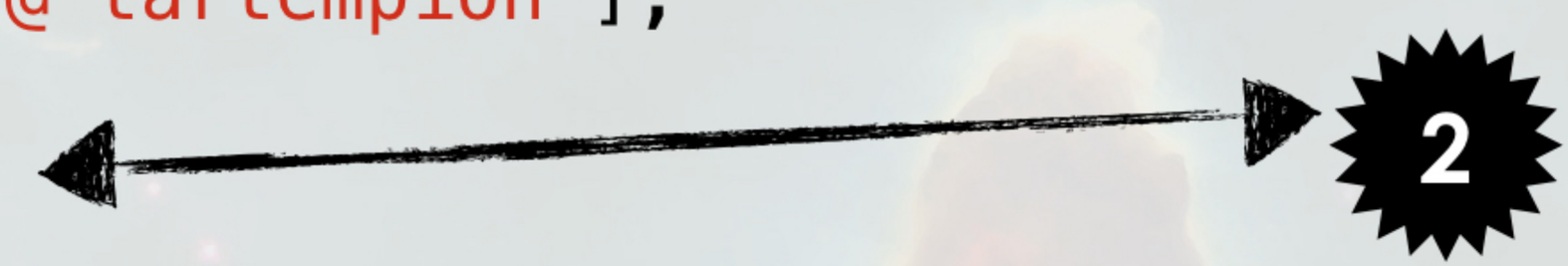

```
Student *aStudent;

aStudent = [[Student alloc]
                initWithNumber:2301666];
[aStudent setName:@"tartempion"];

[aStudent retain];

NSLog(@"Nouvel étudiant : %@",
      [aStudent identity]);

[aStudent release];

[aStudent release];
```



F. Kordon - Sorbonne Université - CC2018

An example

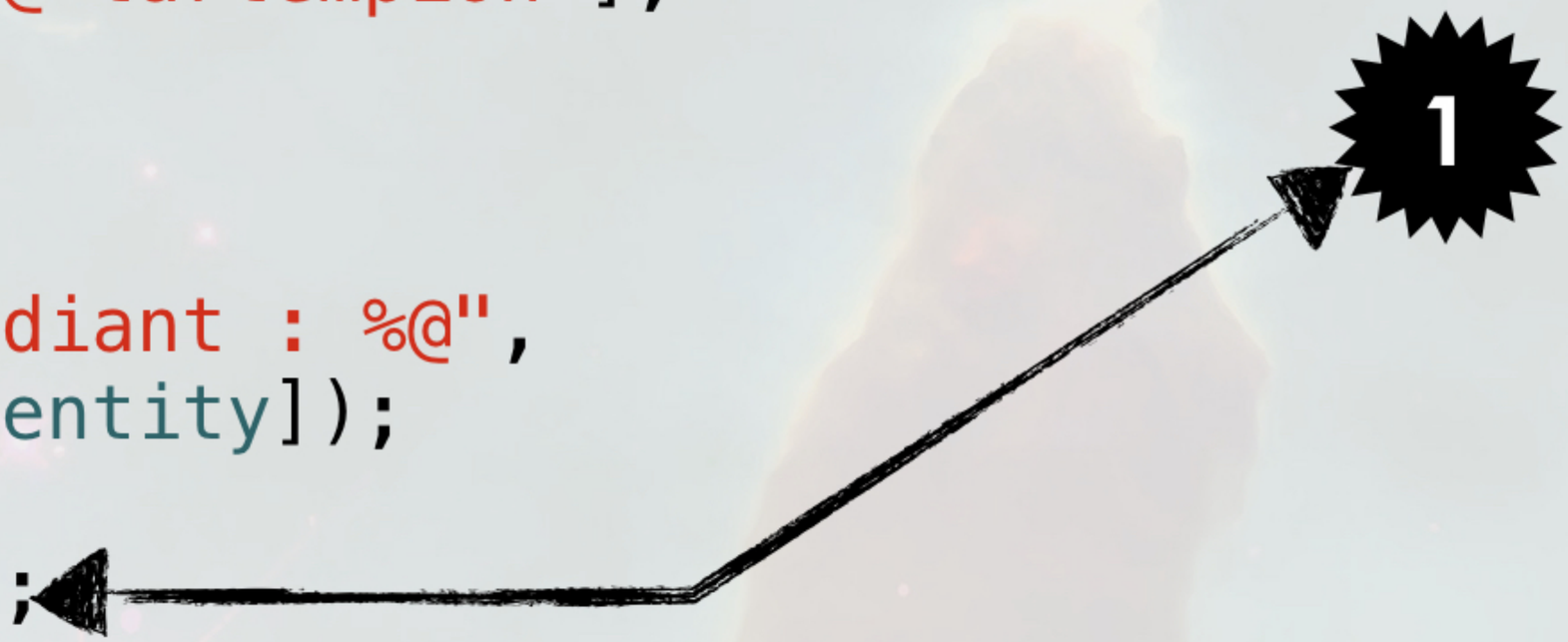

```
Student *aStudent;

aStudent = [[Student alloc]
                initWithNumber:2301666];
[aStudent setName:@"tartempion"];

[aStudent retain];

NSLog(@"Nouvel étudiant : %@",
      [aStudent identity]);

[aStudent release];

[aStudent release];
```



F. Kordon - Sorbonne Université - CC2018

An example

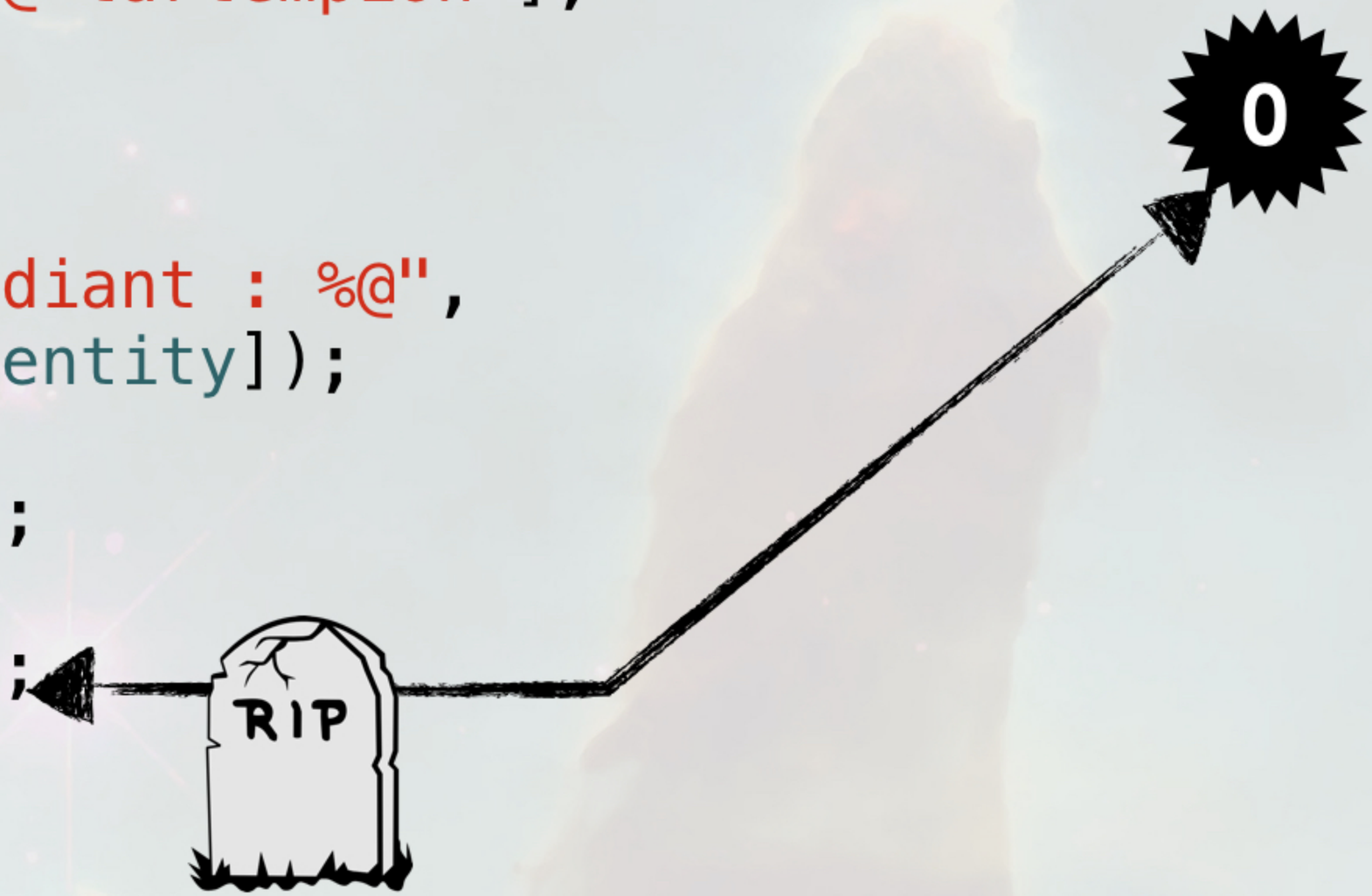

```
Student *aStudent;

aStudent = [[Student alloc]
                initWithNumber:2301666];
[aStudent setName:@"tartempion"];

[aStudent retain];

NSLog(@"Nouvel étudiant : %@",
      [aStudent identity]);

[aStudent release];

[aStudent release];
```



F. Kordon - Sorbonne Université - CC2018

An example


```
Student *aStudent;

aStudent = [[Student alloc]
                initWithNumber:2301666];
[aStudent setName:@"tartempion"];

[aStudent retain];

NSLog(@"Nouvel étudiant : %@",
      [aStudent identity]);

[aStudent release];

[aStudent release];
```



F. Kordon - Sorbonne Université - CC2018

Talk with ghosts?

Classical problem

```
Student *anotherStudent = [[Student alloc] init];
// ...
[anotherStudent release];
NSLog([anotherStudent identity]); // CRASH !
```

Good practice (to use)

```
Student *oneMoreStudent = [[Student alloc] init];
// ...
[oneMoreStudent release];
oneMoreStudent = nil;

// OK (you may talk to ghosts)
NSLog([oneMoreStudent identity]);
```

Links between objects

Two related students

```
#import "Student.h"

@implementation Student

Student *myCoworker; // a co-worker

- (id) init {...}

- (id) initWithNumber:(int)aNumber {...}

- (id) initWithNumber:(int)aNumber andName:(NSString*)aName {...}

- (NSString*) identity {...}

- (void) dealloc {...}

- (void) setCoworker:(Student*)aCoworker {
    if (myCoworker != aCoworker) {

        myCoworker = aCoworker; // the "assign" way

    }
}
@end
```

F. Kordon - Sorbonne Université - CC2018

Links between objects

Two related students

```
#import "Student.h"

@implementation Student

Student *myCoworker; // a co-worker

- (id) init {...}

- (id) initWithNumber:(int)aNumber {...}

- (id) initWithNumber:(int)aNumber andName:(NSString*)aName {...}

- (NSString*) identity {...}

- (void) dealloc {...}

- (void) setCoworker:(Student*)aCoworker {
    if (myCoworker != aCoworker) {
        [myCoworker release]; // dropping (co)ownership
        myCoworker = aCoworker; // the "assign" way


    }
}
@end
```

F. Kordon - Sorbonne Université - CC2018

Links between objects

Two related students

```
#import "Student.h"

@implementation Student

Student *myCoworker; // a co-worker

- (id) init {...}

- (id) initWithNumber:(int)aNumber {...}

- (id) initWithNumber:(int)aNumber andName:(NSString*)aName {...}

- (NSString*) identity {...}

- (void) dealloc {...}

- (void) setCoworker:(Student*)aCoworker {
    if (myCoworker != aCoworker) {
        [myCoworker release]; // dropping (co)ownership

        aCoworker = [aCoworker retain]; // be a (co)owner
    }
}
@end
```

F. Kordon - Sorbonne Université - CC2018

Behavior classification

Strong liaisons

- The «owner» disappears ⇒ the «owned» disappears too

Weak liaisons

- Independence between «owner» and «owned»

Discussed later

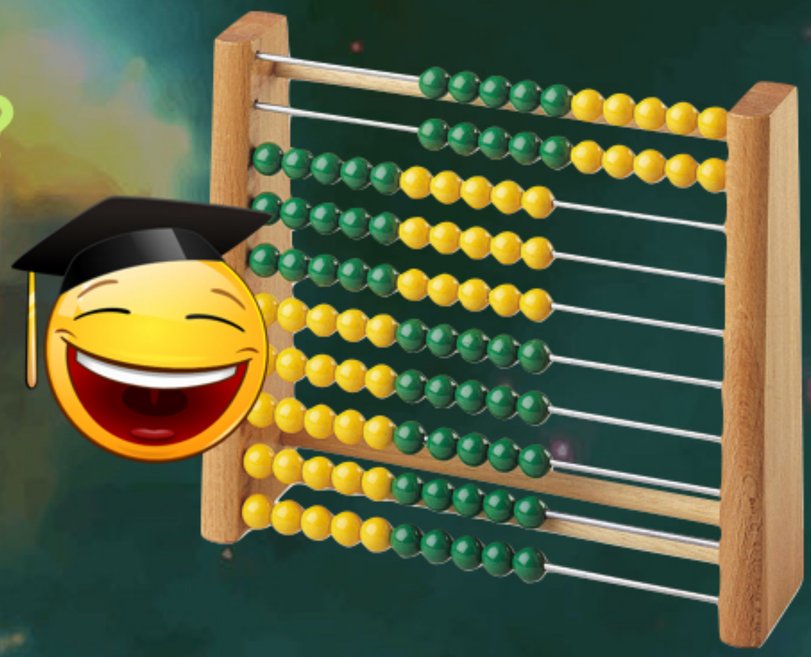
In a future video

F. Kordon - Sorbonne Université - CC2018

As a conclusion…

Is it really difficult?

- You need to count



Once again, a common technique

- Used in environment «memory sensitive»
  - **Real time environment (efficiency of garbage collection)**
- Main advantage
  - **Passive mechanism (on the contrary to memory traversal)**

F. Kordon - Sorbonne Université - CC2018