Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Mobile development 3

Report TD 3 :

Jonathan Stoppani et Elias Medawar

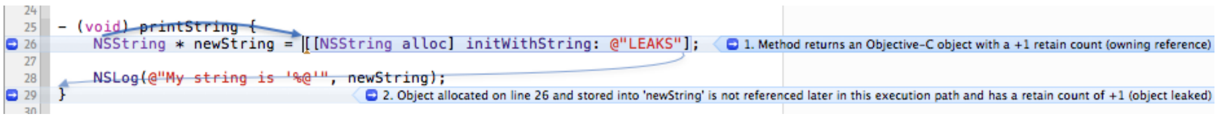
Version: March 15, 2011

1 Memory

1.1 Memory leaks

By natively coding the requested methods and calling them without particular attention, different memory leaks will be created. Some of them are correctly recognized by the XCode Analyzer but some are not.

The recognized leaks are represented in the images below:

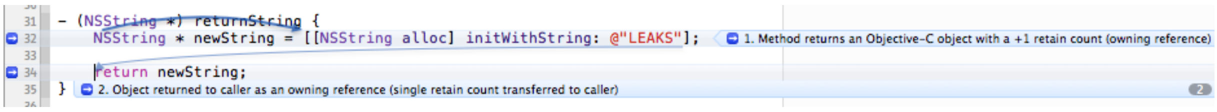


```
- (void) printString {
    NSString * newString = [[NSString alloc] initWithString: @"LEAKS"];

    NSLog(@"My string is '%@'", newString);
}
```

1. Method returns an Objective-C object with a +1 retain count (owning reference)

2. Object allocated on line 26 and stored into 'newString' is not referenced later in this execution path and has a retain count of +1 (object leaked)

Newly created string is not correctly released before exiting the method.

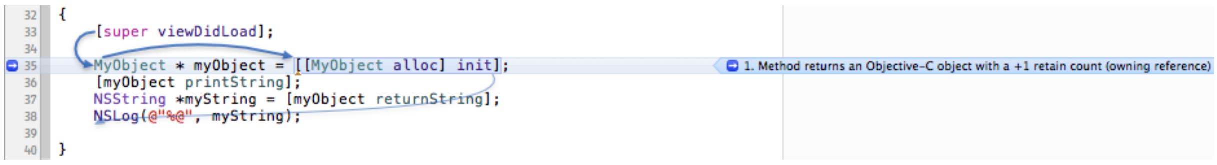


```
- (NSString *) returnString {
    NSString * newString = [[NSString alloc] initWithString: @"LEAKS"];

    return newString;
}
```

1. Method returns an Objective-C object with a +1 retain count (owning reference)

2. Object returned to caller as an owning reference (single retain count transferred to caller)

Returned string should be autoreleased before exiting the method.

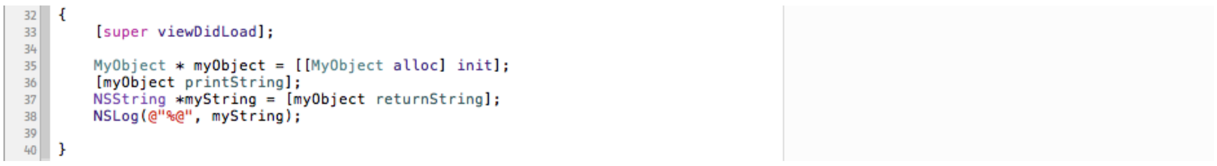


```
{
    [super viewDidLoad];

    MyObject * myObject = [[MyObject alloc] init];
    [myObject printString];
    NSString *myString = [myObject returnString];
    NSLog(@"%@", myString);

}
```

1. Method returns an Objective-C object with a +1 retain count (owning reference)

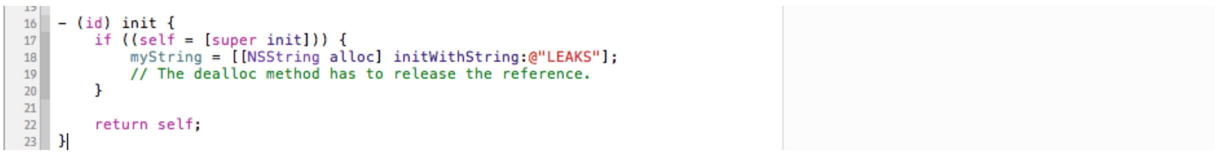
The newly allocated object is never released.

There are other cases where the Analayser was not able to detect a memory leak, such as in the following examples:



```
{
    [super viewDidLoad];

    MyObject * myObject = [[MyObject alloc] init];
    [myObject printString];
    NSString *myString = [myObject returnString];
    NSLog(@"%@", myString);

}
```

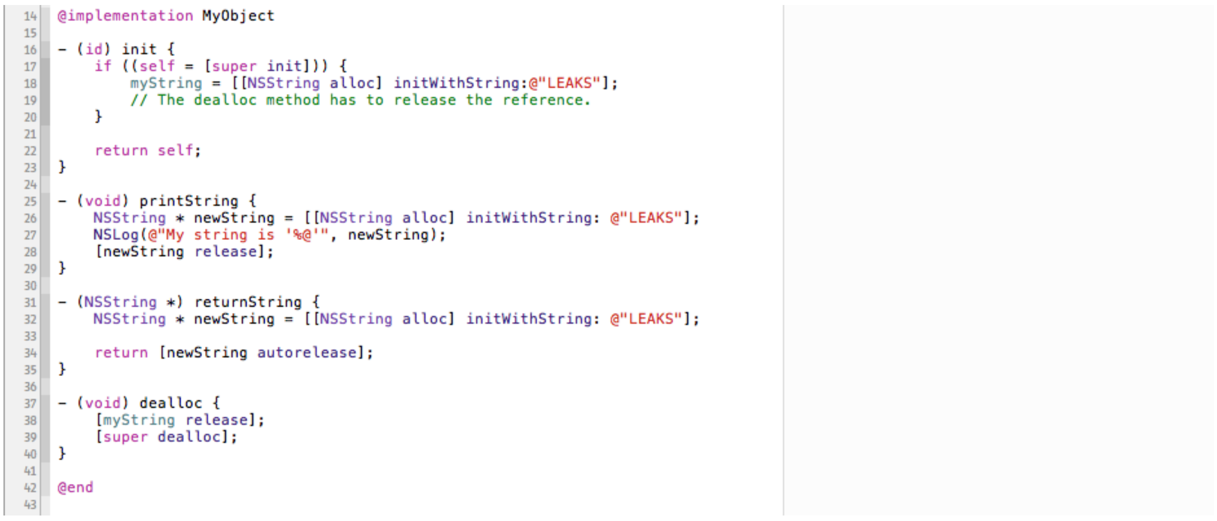
The analyzer assumes that the string returned by the returnString method has already been autoreleased, but it is not the case.



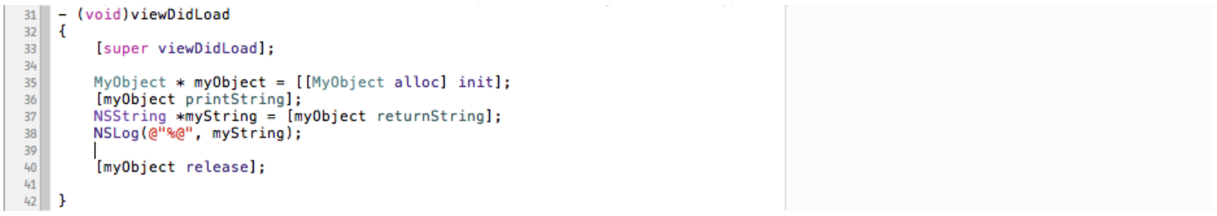
```
- (id) init {
    if ((self = [super init])) {
        myString = [[NSString alloc] initWithString:@"LEAKS"];
        // The dealloc method has to release the reference.
    }

    return self;
}
```

The string created in the init method is never released because the object does not have a dealloc method.

The correct – leak free – source code is represented below:



```
@implementation MyObject

- (id) init {
    if ((self = [super init])) {
        myString = [[NSString alloc] initWithString:@"LEAKS"];
        // The dealloc method has to release the reference.
    }

    return self;
}

- (void) printString {
    NSString * newString = [[NSString alloc] initWithString: @"LEAKS"];
    NSLog(@"My string is '%@'", newString);
    [newString release];
}

- (NSString *) returnString {
    NSString * newString = [[NSString alloc] initWithString: @"LEAKS"];

    return [newString autorelease];
}

- (void) dealloc {
    [myString release];
    [super dealloc];
}

@end
```



```
- (void)viewDidLoad
{
    [super viewDidLoad];

    MyObject * myObject = [[MyObject alloc] init];
    [myObject printString];
    NSString *myString = [myObject returnString];
    NSLog(@"%@", myString);
    |
    [myObject release];

}
```

Notice that the string returned by the returnString method is never retained and released on the caller part. This behaviour is still correct (although no a best-practice) and explained below.

1.2 Autorelease pools

When an object is autoreleased, it is registered to the innermost autorelease pool (in our case the pool created in the main method) and persisted in memory until the autorelease pool is not drained or released.

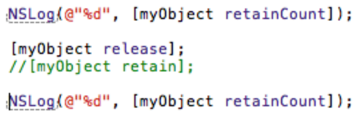
In the previous example, this means that the autoreleased string returned by the returnString method is actually kept in memory until the autorelease pool is not released and can be used without an additional retain and release call on it.

More information about this topic can be found on the online help at the following URL: `http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/MemoryMgmt/Articles/mmAutoreleasePool` `html`.

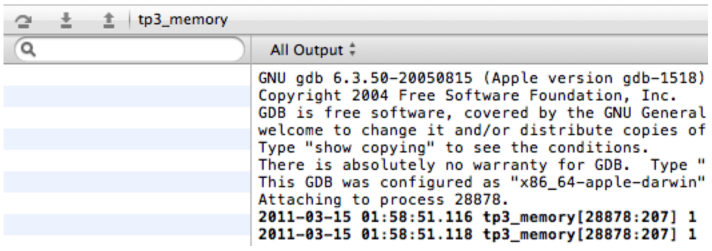
1.3 Retain/release

By adding different retain/release called, we observed the following behavior:

- The first method call on a released object normally works, the second one causes a EXC_BAD_ACCESS error.
- The same holds true even if sleeping 5 seconds after the object release.
- The retainCount on a released object is 1 event when it should be 0 (we were able to retrieve this value thanks to the property above):



```
NSLog(@"%d", [myObject retainCount]);

[myObject release];
//[myObject retain];

NSLog(@"%d", [myObject retainCount]);
```



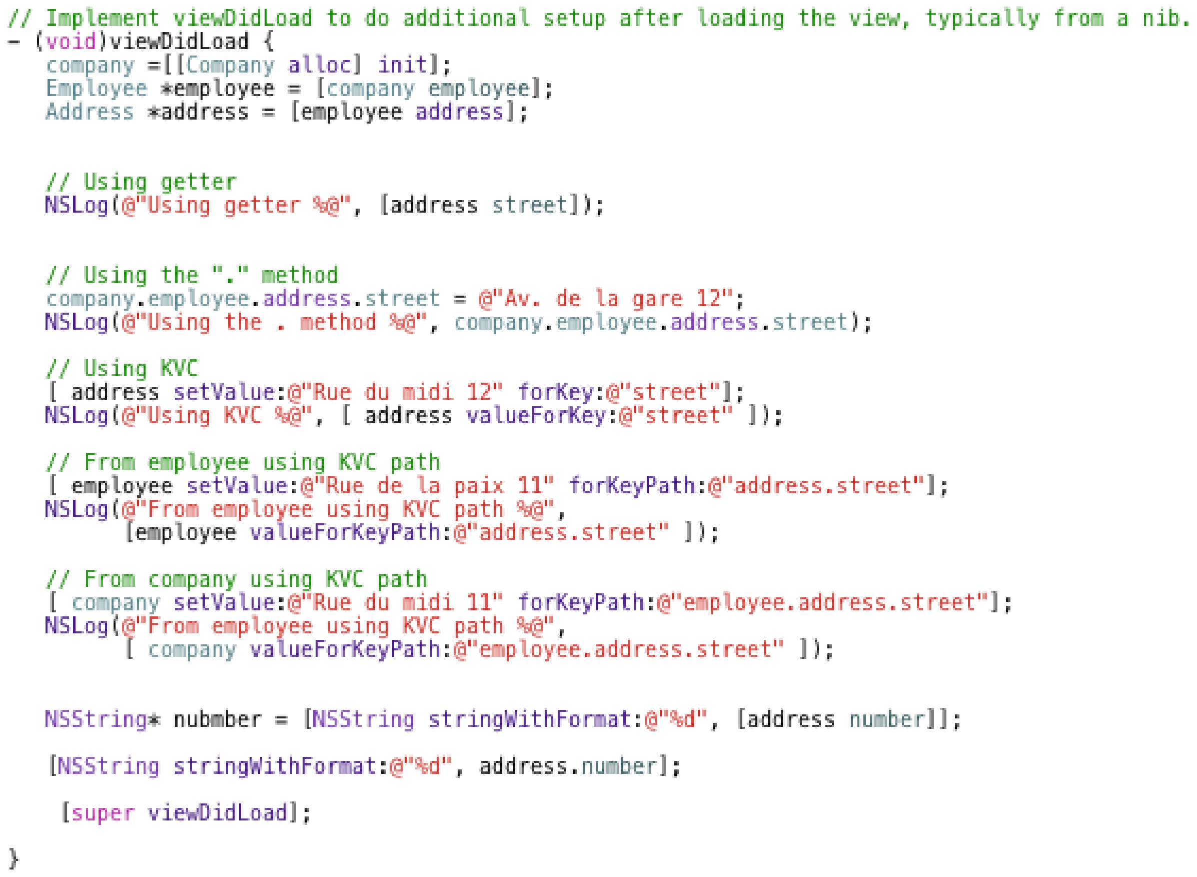
```
GNU gdb 6.3.50-20050815 (Apple version gdb-1518)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General
welcome to change it and/or distribute copies of
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "
This GDB was configured as "x86_64-apple-darwin"
Attaching to process 28878.
2011-03-15 01:58:51.116 tp3_memory[28878:207] 1
2011-03-15 01:58:51.118 tp3_memory[28878:207] 1
```

- It is possible to call the retain method on a just released object but the result is not guaranteed. Sometimes the object is kept in memory but sometimes an EXC_BAD_ACCESS error is raised upon the next call.

By observing the behaviors above, we can conclude that the object is not releaed immediately when its retain count reaches 0 but we were not able to confirm the exact behavior (the documentation and the web-searches led to no results) of the memory releasing process. Anyway, it is not safe to rely on such edge-cases and not-guaranteed behaviors in our code and the common-sense dictates to not use an object after it was released (and – for extra safety – to set it to null after the call to release).

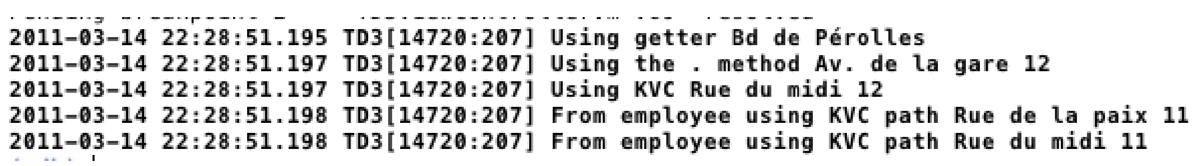
2 KVC

2.1 Source code in address for the KVC "request"



```
// Implement viewDidLoad to do additional setup after loading the view, typically from a nib.
- (void)viewDidLoad {
    company =[[Company alloc] init];
    Employee *employee = [company employee];
    Address *address = [employee address];


    // Using getter
    NSLog(@"Using getter %@", [address street]);


    // Using the "." method
    company.employee.address.street = @"Av. de la gare 12";
    NSLog(@"Using the . method %@", company.employee.address.street);

    // Using KVC
    [ address setValue:@"Rue du midi 12" forKey:@"street"];
    NSLog(@"Using KVC %@", [ address valueForKey:@"street" ]);

    // From employee using KVC path
    [ employee setValue:@"Rue de la paix 11" forKeyPath:@"address.street"];
    NSLog(@"From employee using KVC path %@",
            [employee valueForKeyPath:@"address.street" ]);

    // From company using KVC path
    [ company setValue:@"Rue du midi 11" forKeyPath:@"employee.address.street"];
    NSLog(@"From employee using KVC path %@",
            [ company valueForKeyPath:@"employee.address.street" ]);


    NSString* nubmber = [NSString stringWithFormat:@"%d", [address number]];

    [NSString stringWithFormat:@"%d", address.number];

     [super viewDidLoad];

}
```

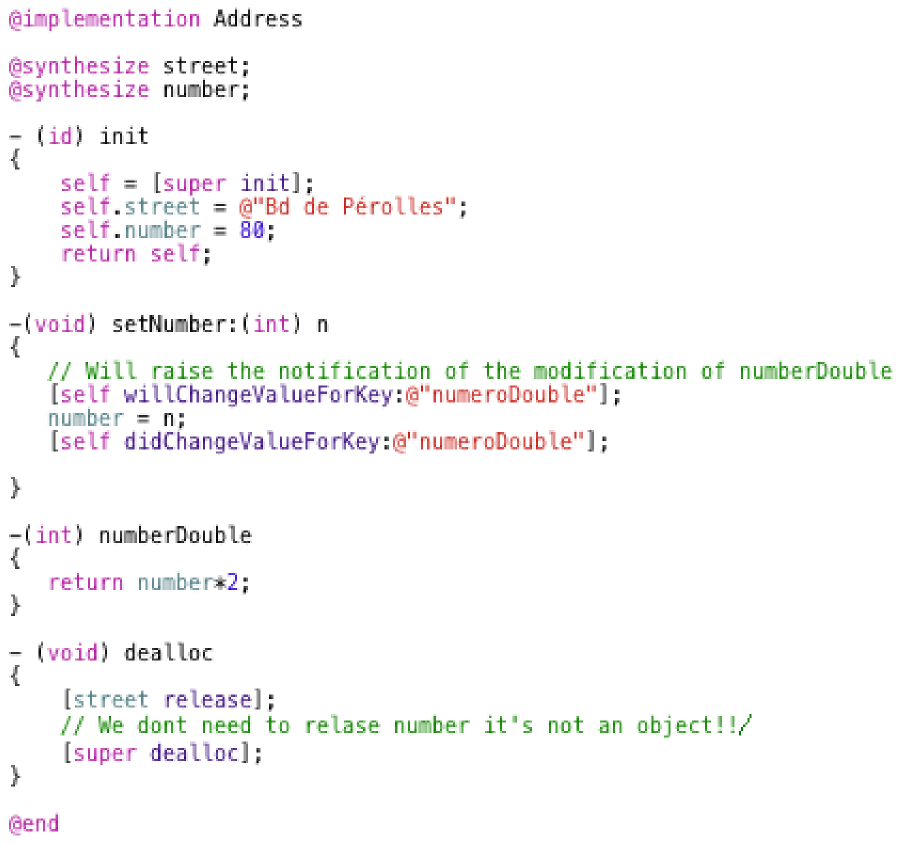
2.2 Output



```
2011-03-14 22:28:51.195 TD3[14720:207] Using getter Bd de Pérolles
2011-03-14 22:28:51.197 TD3[14720:207] Using the . method Av. de la gare 12
2011-03-14 22:28:51.197 TD3[14720:207] Using KVC Rue du midi 12
2011-03-14 22:28:51.198 TD3[14720:207] From employee using KVC path Rue de la paix 11
2011-03-14 22:28:51.198 TD3[14720:207] From employee using KVC path Rue du midi 11
```

3 KVO

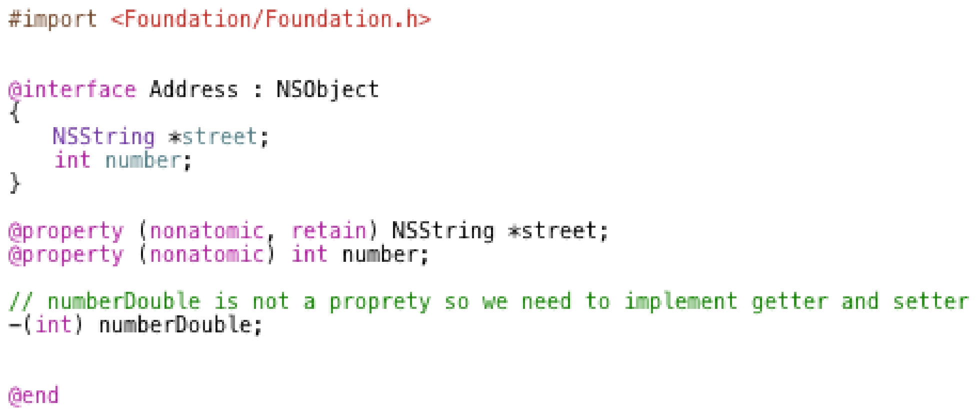
3.1 M file of the address with the int and numberDouble



```
@implementation Address

@synthesize street;
@synthesize number;

- (id) init
{
    self = [super init];
    self.street = @"Bd de Pérolles";
    self.number = 80;
    return self;
}

-(void) setNumber:(int) n
{
   // Will raise the notification of the modification of numberDouble
   [self willChangeValueForKey:@"numeroDouble"];
   number = n;
   [self didChangeValueForKey:@"numeroDouble"];

}

-(int) numberDouble
{
   return number*2;
}

- (void) dealloc
{
    [street release];
    // We dont need to relase number it's not an object!!/
    [super dealloc];
}

@end
```

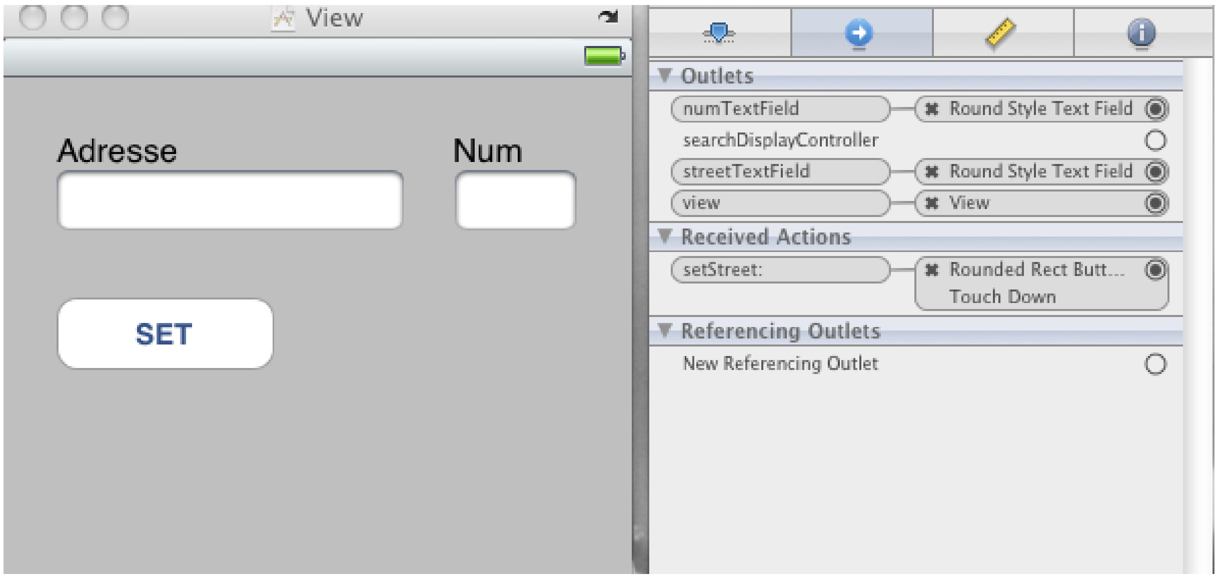
The numberDouble is a calculated value so we have to raise "manually " the notifications.

3.2 H file of the address with the int and numberDouble



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


@interface Address : NSObject
{
    NSString *street;
    int number;
}

@property (nonatomic, retain) NSString *street;
@property (nonatomic) int number;

// numberDouble is not a proprety so we need to implement getter and setter
-(int) numberDouble;


@end
```

3.3 UI with 2 text-field and a button


View
Adresse
Num
SET
Outlets
numTextField — Round Style Text Field
searchDisplayController
streetTextField — Round Style Text Field
view — View
Received Actions
setStreet: — Rounded Rect Butt... Touch Down
Referencing Outlets
New Referencing Outlet

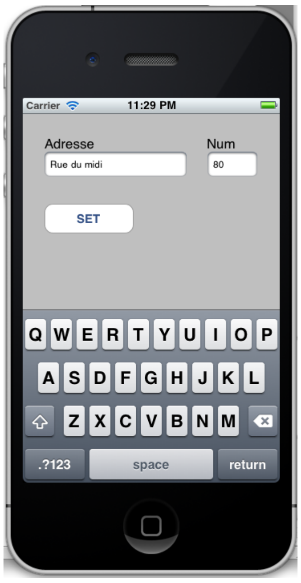
Here we see the creation and links in **Interface builder**

3.4 Setting value of text fields

```
- (void)viewDidLoad {
    ...

    //Transform int to string
    NSString* nubmber = [NSString stringWithFormat:@"%d", [address number]];
    streetTextField.text = address.street; // Display values
    numTextField.text = nubmber;

     [super viewDidLoad];

}
```

At the load of the application we set the value that will be displayed
The conversion form int to string was found on this website: http://stackoverflow.com/questions/169925/how-to-do-string-conversions-in-objective-c

3.5 Result after loading

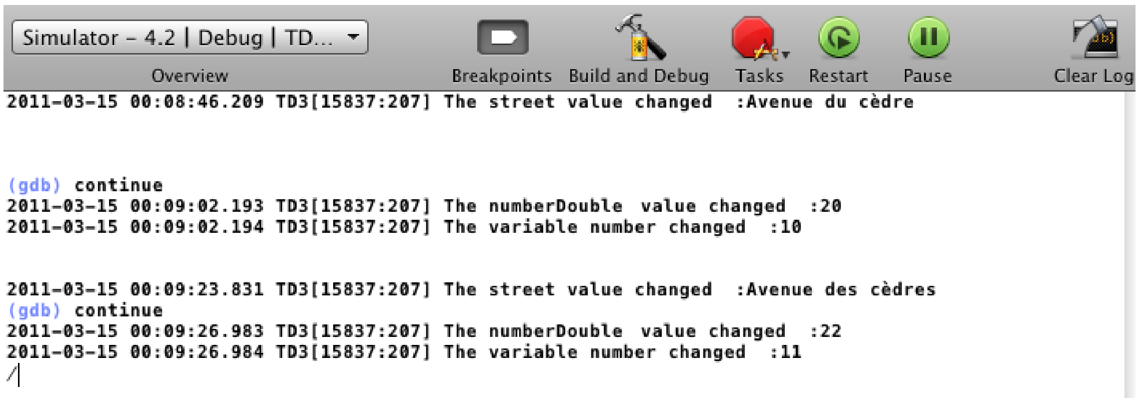


3.6 Registering and receiving the KVO notification in the employee class

```
#import "Employee.h"
#import "Address.h"

@implementation Employee

@synthesize address;

-(id) init
{
    self = [super init];

    address = [[Address alloc] init];

    // Register the KVO notification
    [address addObserver:self forKeyPath:@"street" options:0 context:NULL];
    [address addObserver:self forKeyPath:@"number" options:0 context:NULL];
    [address addObserver:self forKeyPath:@"numberDouble" options:0 context:NULL];

    return self;/
}

// This method will be called on changes
- (void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object
                        change:(NSDictionary *)change context:(void *)context
{
    if([keyPath isEqualToString:@"number"])
       NSLog(@"The variable number changed  :%@", [object valueForKeyPath:keyPath]);
    else if([keyPath isEqualToString:@"street"])
       NSLog(@"The street value changed  :%@", [object valueForKeyPath:keyPath]);
    else if([keyPath isEqualToString:@"numberDouble"])
       NSLog(@"The numberDouble   value changed  :%@", [object valueForKeyPath:keyPath]);
}

-(void) dealloc
{
    [address removeObserver:self forKeyPath:@"street"];
    [address removeObserver:self forKeyPath:@"numero"];
    [address release];

    [super dealloc];
}

@end
```

We are responsible to unsubscribe the notification in the dealloc method.

3.7 Result of the execution



```
2011-03-15 00:08:46.209 TD3[15837:207] The street value changed  :Avenue du cèdre


(gdb) continue
2011-03-15 00:09:02.193 TD3[15837:207] The numberDouble  value changed  :20
2011-03-15 00:09:02.194 TD3[15837:207] The variable number changed  :10


2011-03-15 00:09:23.831 TD3[15837:207] The street value changed  :Avenue des cèdres
(gdb) continue
2011-03-15 00:09:26.983 TD3[15837:207] The numberDouble  value changed  :22
2011-03-15 00:09:26.984 TD3[15837:207] The variable number changed  :11
/|
```

1 .For this tests , we change at first the street value then press set
2 The we change the number value and press set
3 Finally we change both value and press set

We tried to call just the didChangeValueForKey in the setNumber method of the address class.But the notification doesn't work without both call wil and didChangeValueForKey.

4 Notifications

4.1 M file of the company class

```
#import "Company.h"
#import "Employee.h"

@implementation Company

@synthesize employee;

- (id) init
{
    self = [super init];

    employee = [[Employee alloc] init];

    // ADD the observer to the Notificationcenter

    [[NSNotificationCenter defaultCenter] addObserver:self selector: @selector(streetChanged:)
                                                 name: @"streetChanged" object: nil];

    return self;
}

// The calback method of the notification
-(void) streetChanged:(NSNotification *)notification
{

    NSLog(@"The streetChanged notification was raised.");
}

- (void) dealloc
{
     [employee release];

     [super dealloc];
}


@end
```

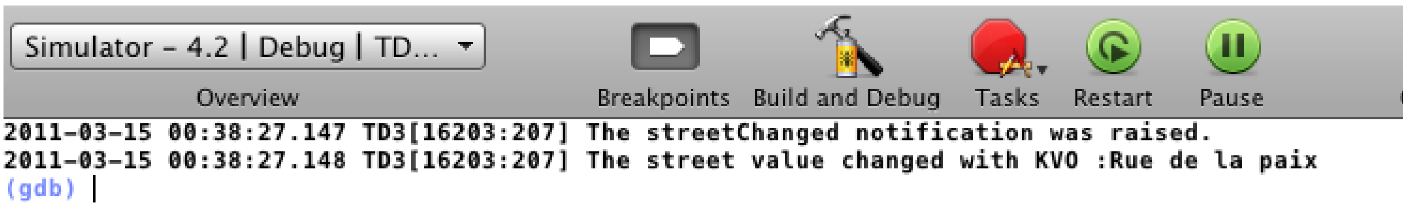
Here we have added an observer and the callback function to recive the specific information.

4.2 Raising notification

```
-(void) setStreet:(NSString*) s
{
   if(![s isEqualToString: street ]){
      // Will raise the notification of the modification of numberDouble
      street = s;
      [[NSNotificationCenter defaultCenter] postNotificationName: @"streetChanged" object:nil];
   }
}
```

In the address class when we change the value of the street we have to post a notification

4.3 Result



```
2011-03-15 00:38:27.147 TD3[16203:207] The streetChanged notification was raised.
2011-03-15 00:38:27.148 TD3[16203:207] The street value changed with KVO :Rue de la paix
(gdb) |
```

The result in the console.

5 Conclusion

KVC is very powerful but we have had some problem with the spelling of variable name. And this errors was not detected at the compilation time but at the run time.
All the sources and documentation are available on gitHub at this address:
https://github.com/eia-fr/mobileDev/tree/master/TD3